

Figure 1 (Prior Art)

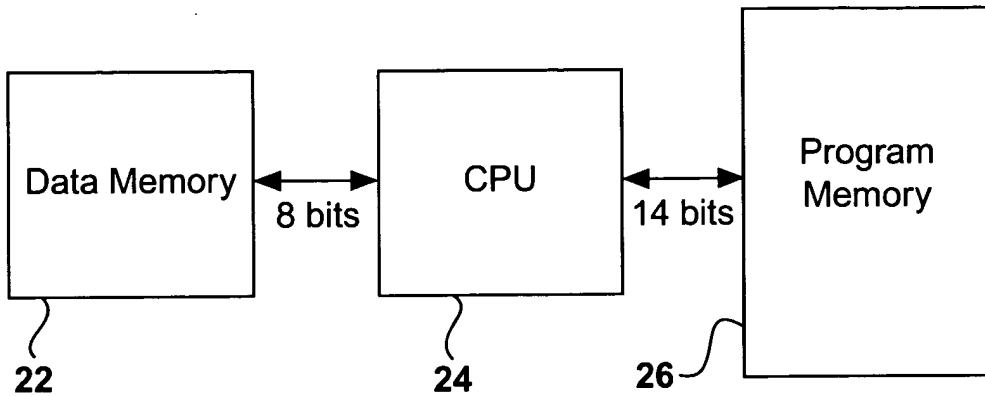


Figure 2 (Prior Art)

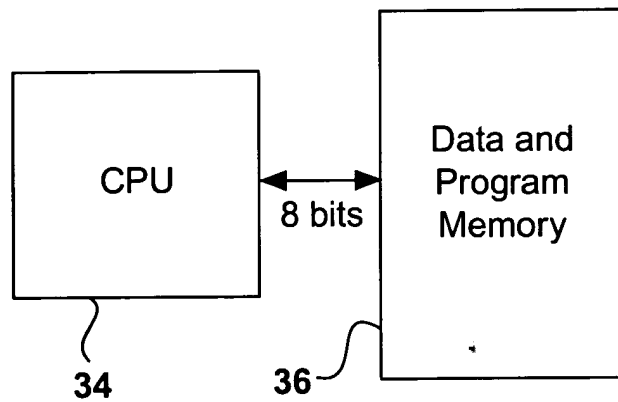


Figure 3 (Prior Art)

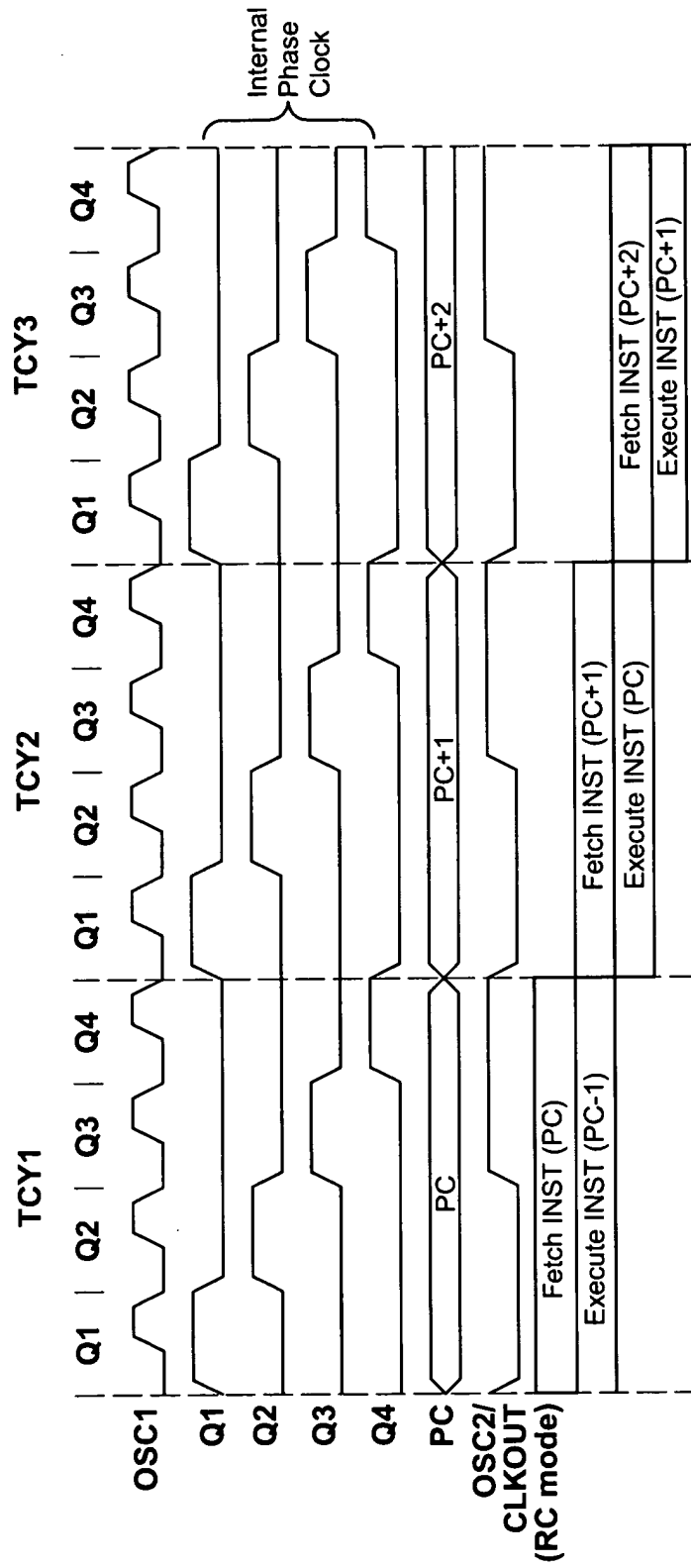


Figure 4 (Prior Art)

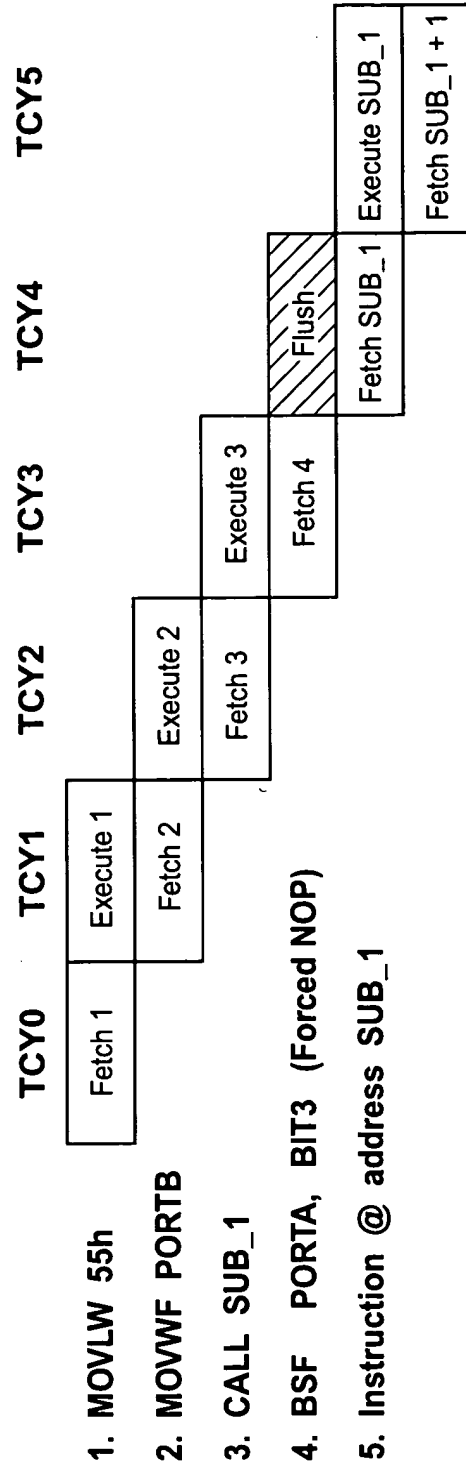


Figure 5 (Prior Art)

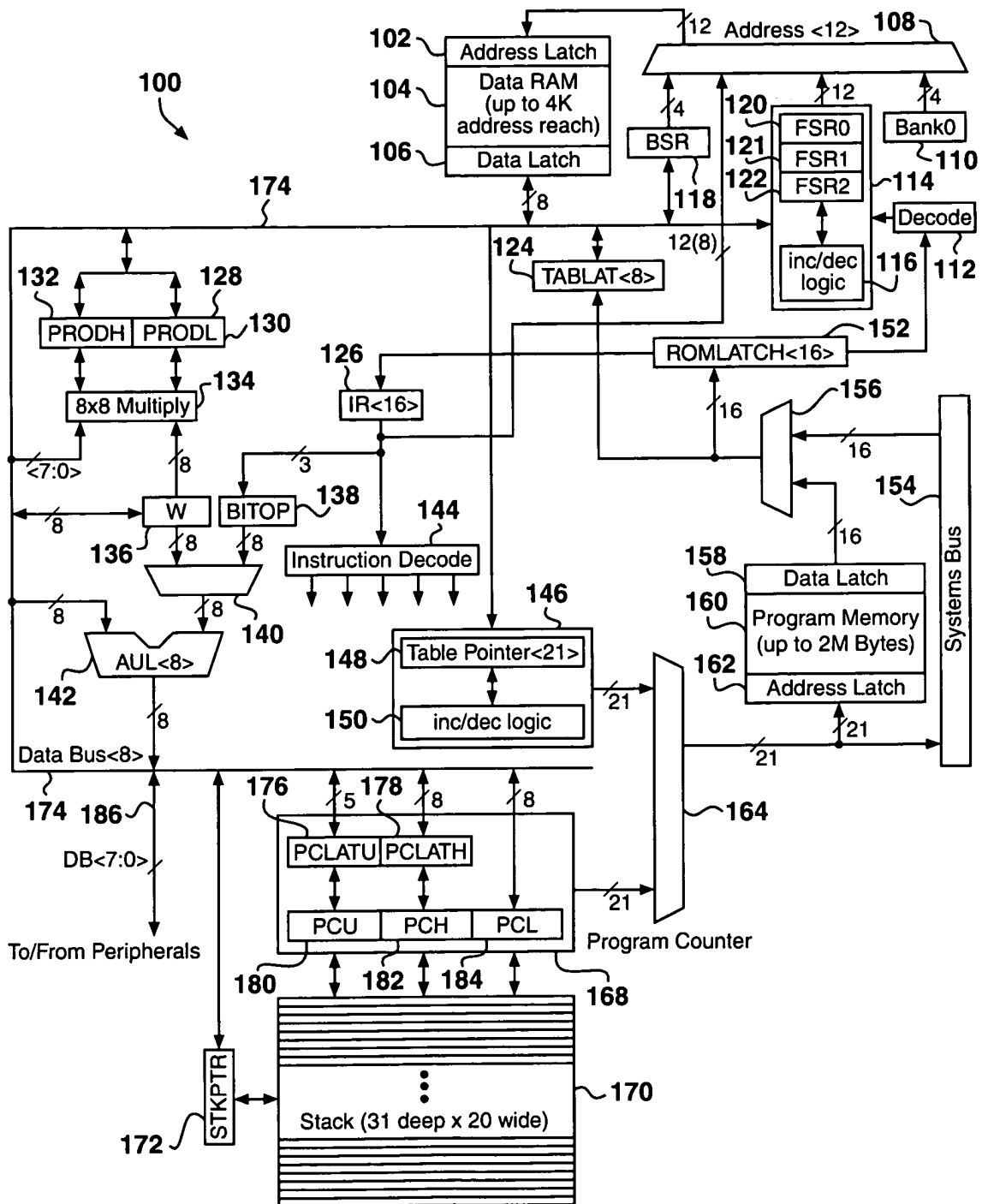
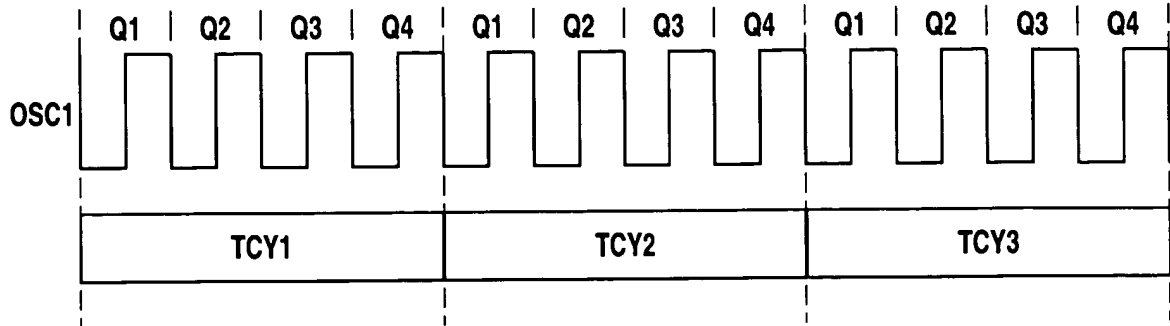
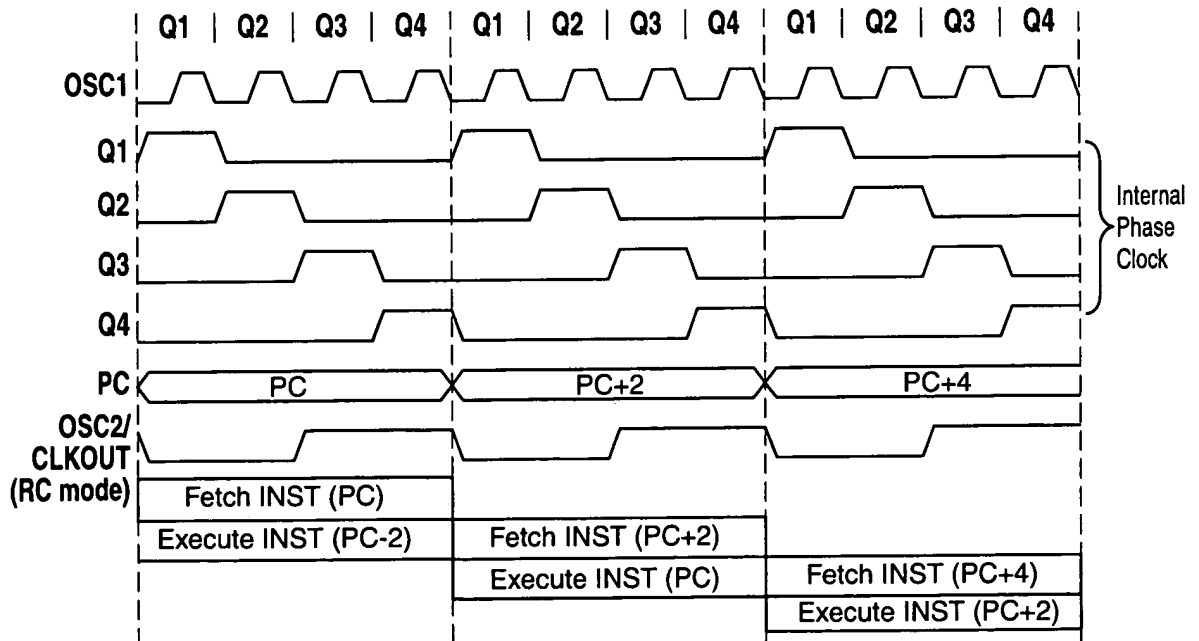


Figure 6

**Figure 7****Figure 8**

7/95

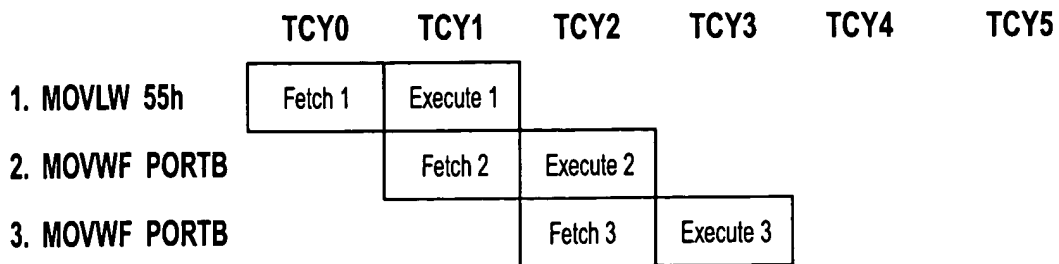


Figure 9

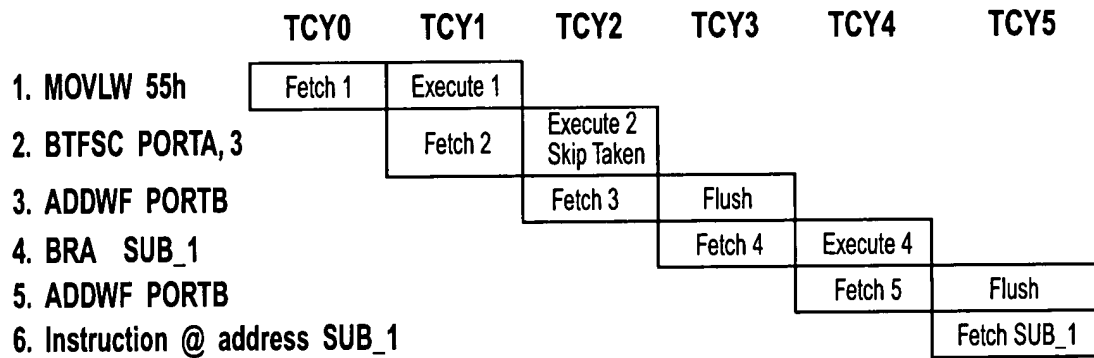


Figure 10

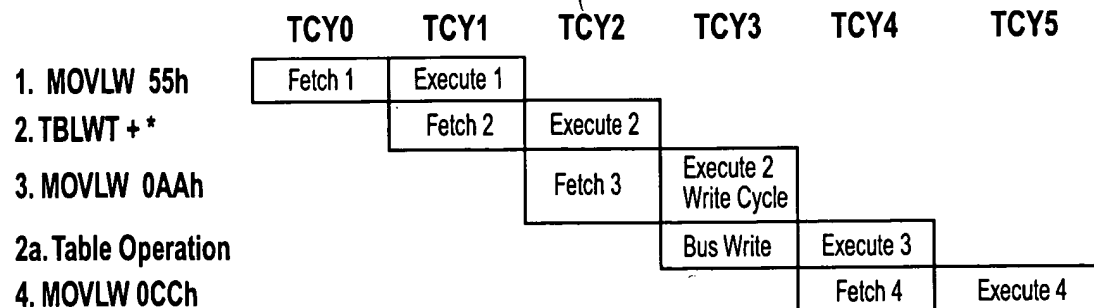


Figure 11

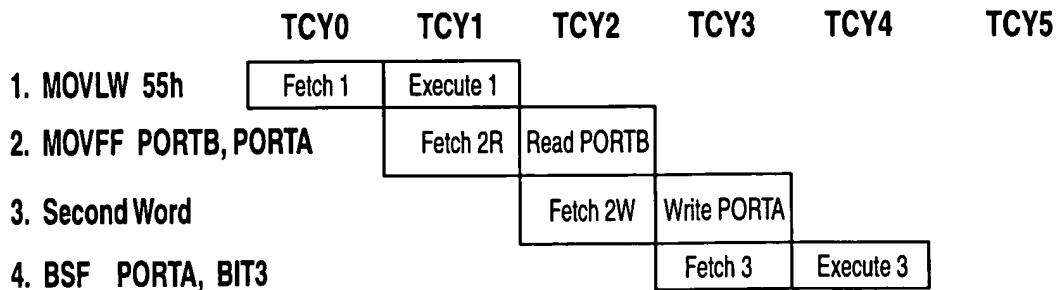


Figure 12

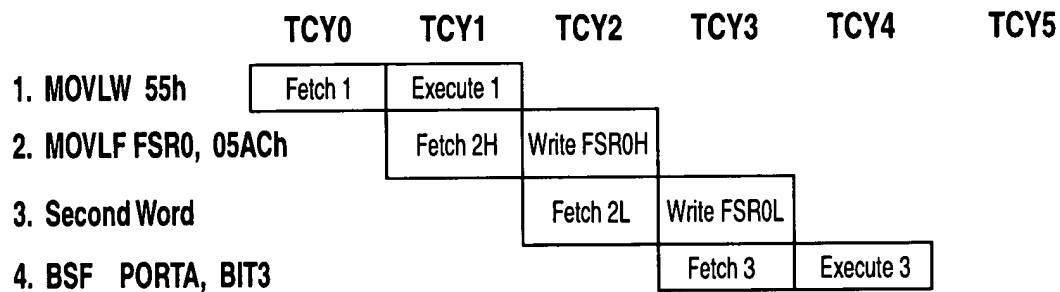


Figure 13

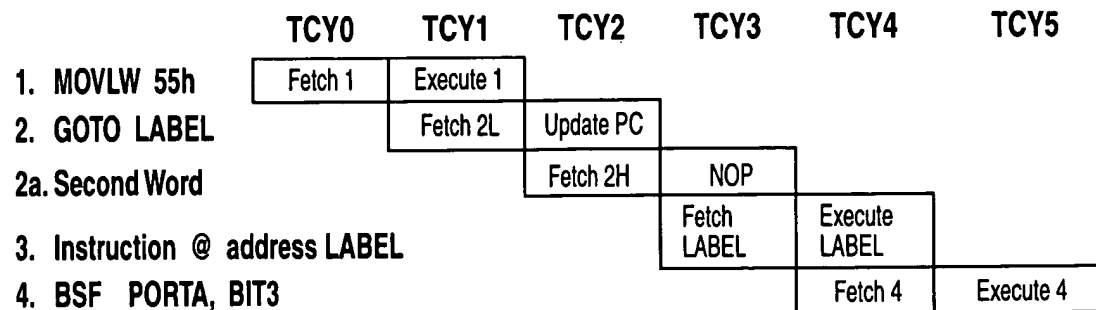
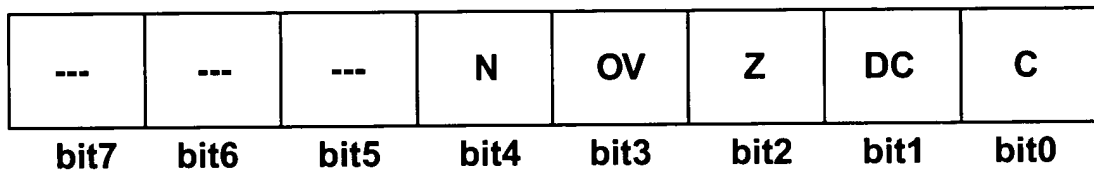


Figure 14

***Figure 15***

10/95

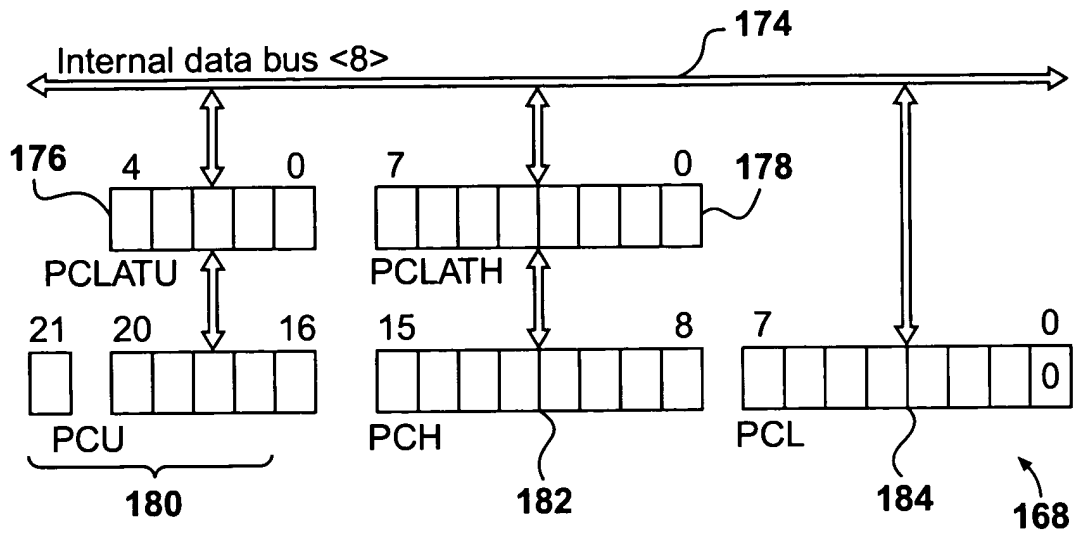


Figure 16

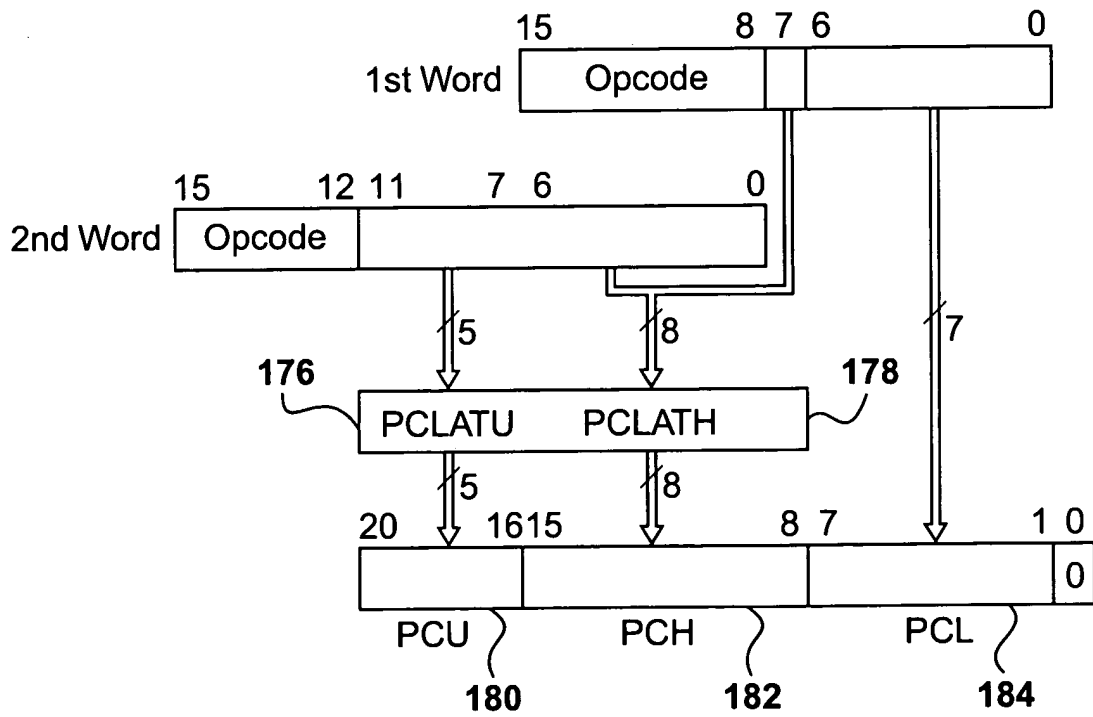
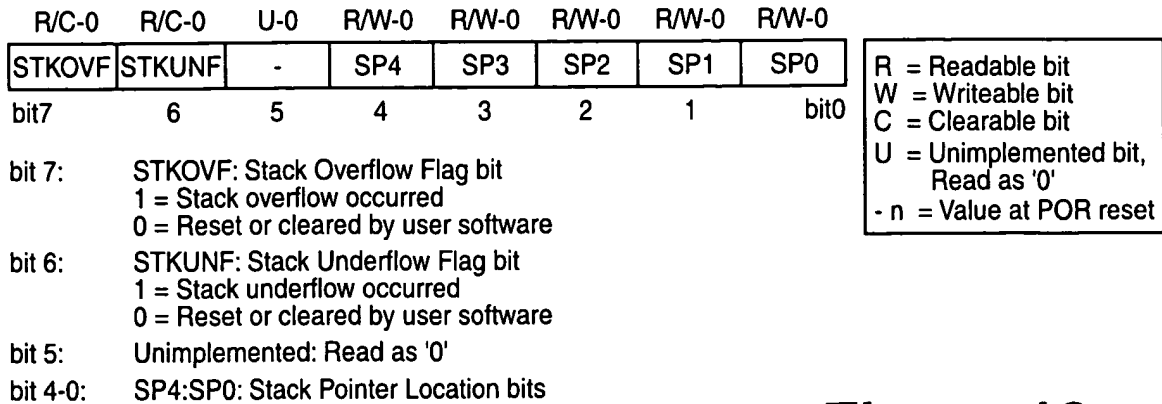


Figure 17

11/95



STKPTR - Stack Pointer Register

Figure 18

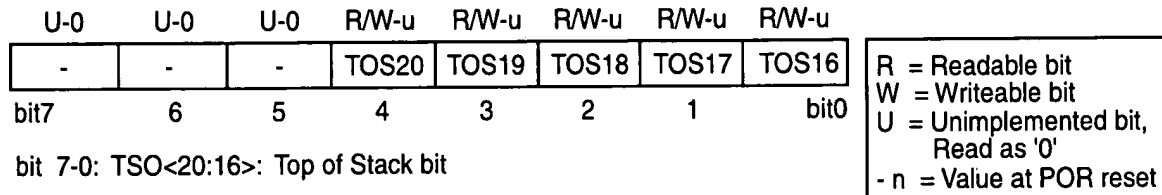


Figure 19

TOSU - Top of Stack Upper

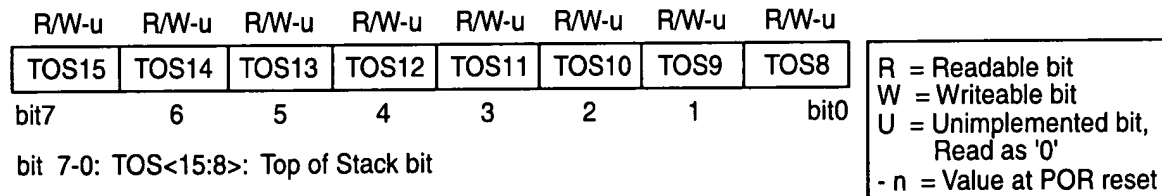


Figure 20

TOSH - Top of Stack High

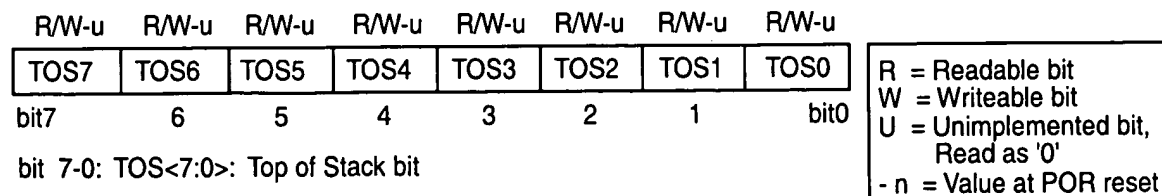
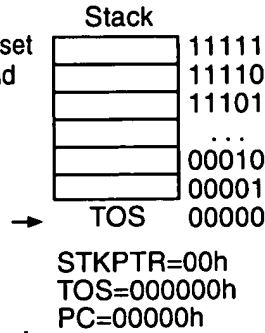


Figure 21

TOSL - Top of Stack Low

12/95

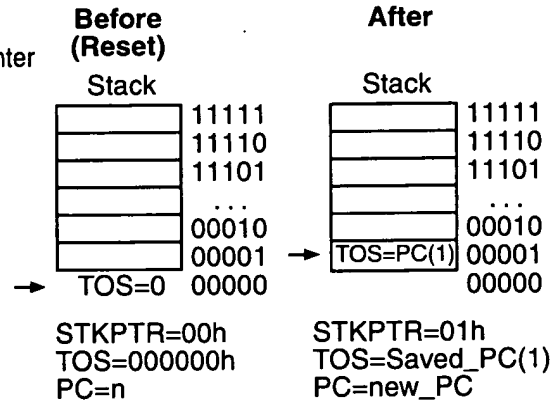
When a device is reset, the PC is loaded with the reset vector (0h). The stack pointer is initialized to 00h, and the Top of Stack register (TOS) is 000000h.



Reset

Figure 22

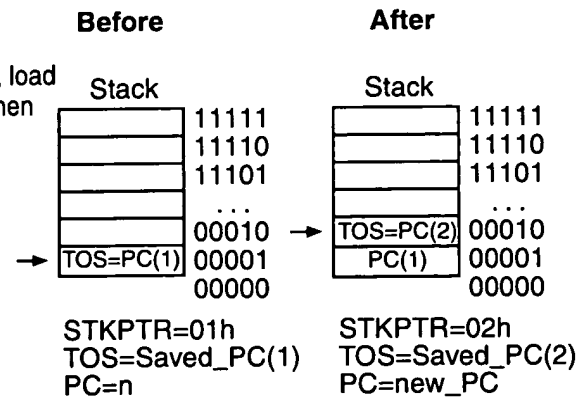
The first push of the stack increments the stack pointer to point to location 1. The value in the PC is loaded into stack level 1. The PC is then updated.



First CALL on an initialized stack

Figure 23

The second push will increment the stack pointer, load the TOS register with the current PC value, and then update the PC.

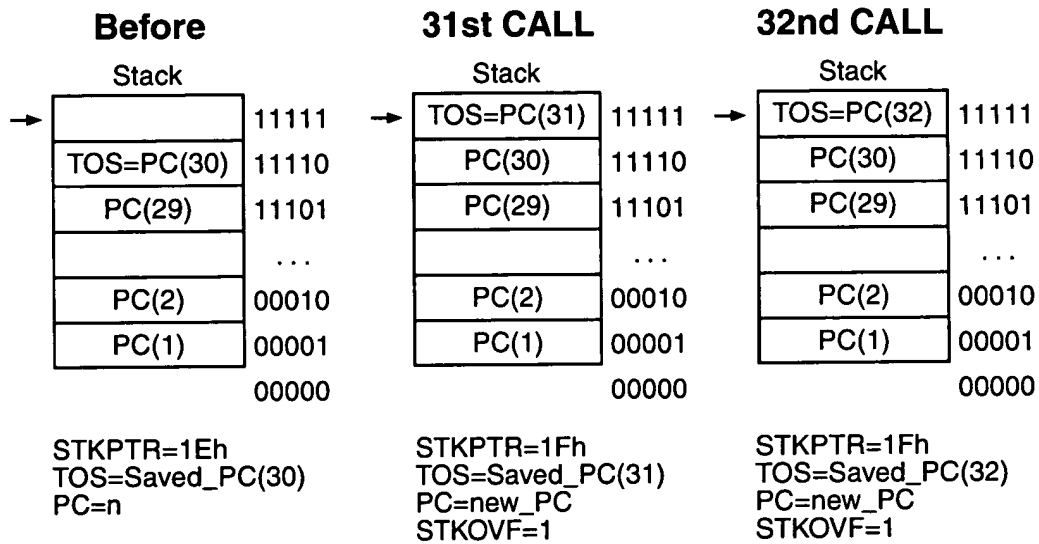


Second Consecutive CALL

Figure 24

The 31st CALL will increment the stack pointer, load the TOS register with the current PC value, and then update the PC. The STKOVF bit is set to indicate the impending stack overflow.

The 32nd CALL will attempt to increment the stack pointer. However the stack pointer is now pointing at the upper most stack level and cannot be incremented. The pointer will be loaded with 1Fh (still pointing to stack level 31), and the TOS will be overwritten with the current PC value. The PC will be updated. The stack overflow bit remains set. Another push will yield the same results. Once the pointer had incremented to 1Fh, it cannot be incremented to a higher value, it can only be cleared or decremented.

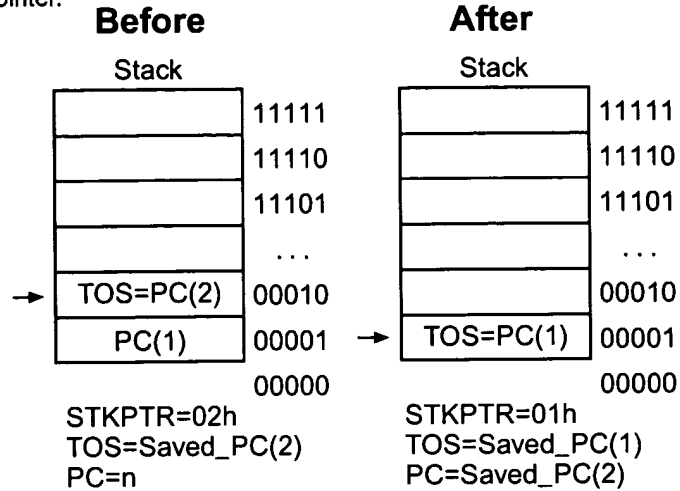


31st and 32nd consecutive CALL

Figure 25

14/95

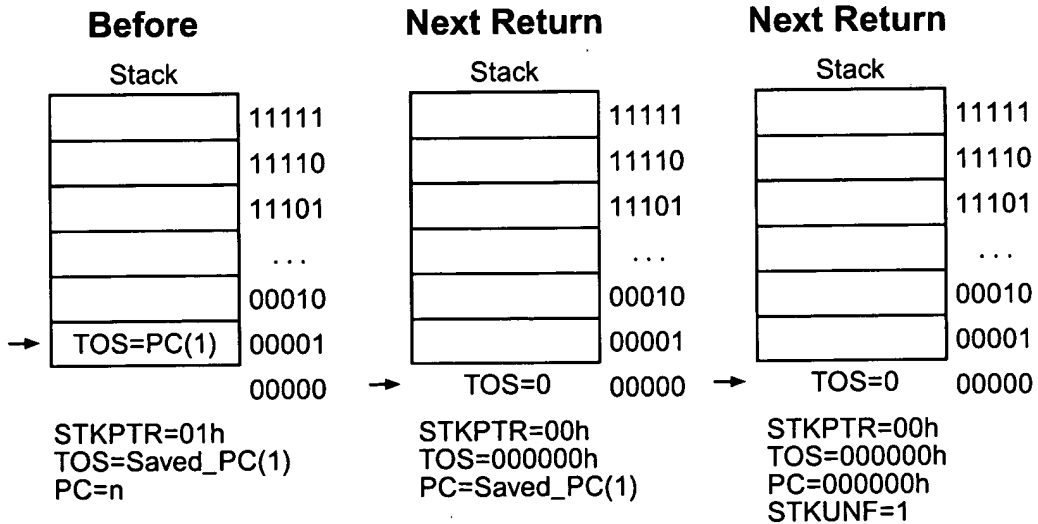
A return pop will copy the contents of the TOS to the PC and then decrement the stack pointer.



Return Pop

Figure 26

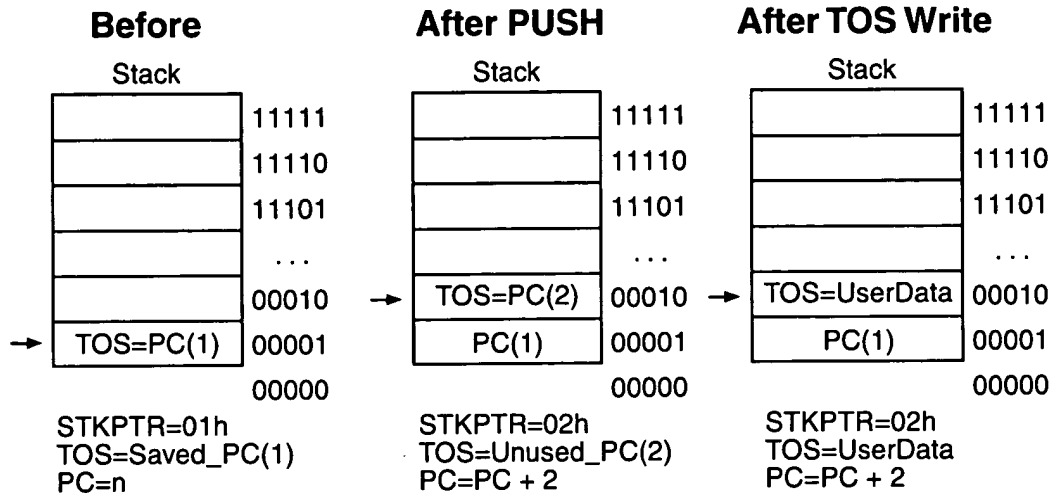
When the stack has been popped enough to reach 00h and the stack pointer can no longer be decremented, further popping will return 000000h to the PC. The stack pointer will maintain the value of 00h. The underflow bit (STKUNF) is set.



Stack Return Pops Causing Stack Underflow

Figure 27

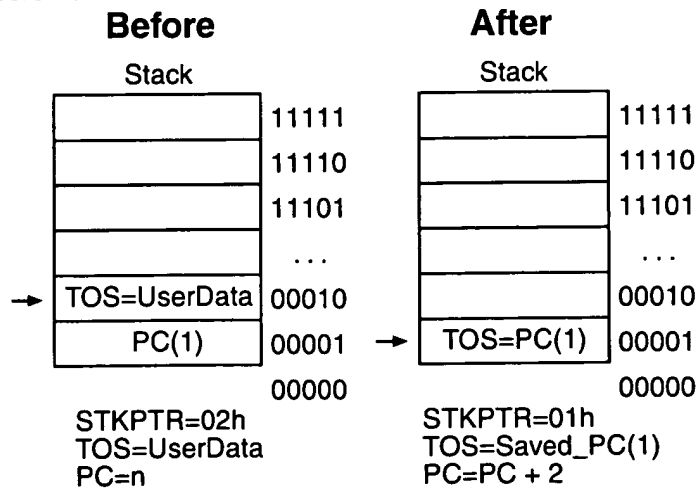
A PUSH instruction performs a similar operation as a call. The PC is incremented to PC+2, the stack pointer is incremented and the TOS is loaded with the PC value (which is essentially a wasted operation). The user will then have access to write values into the TOS registers.



PUSH instruction

Figure 28

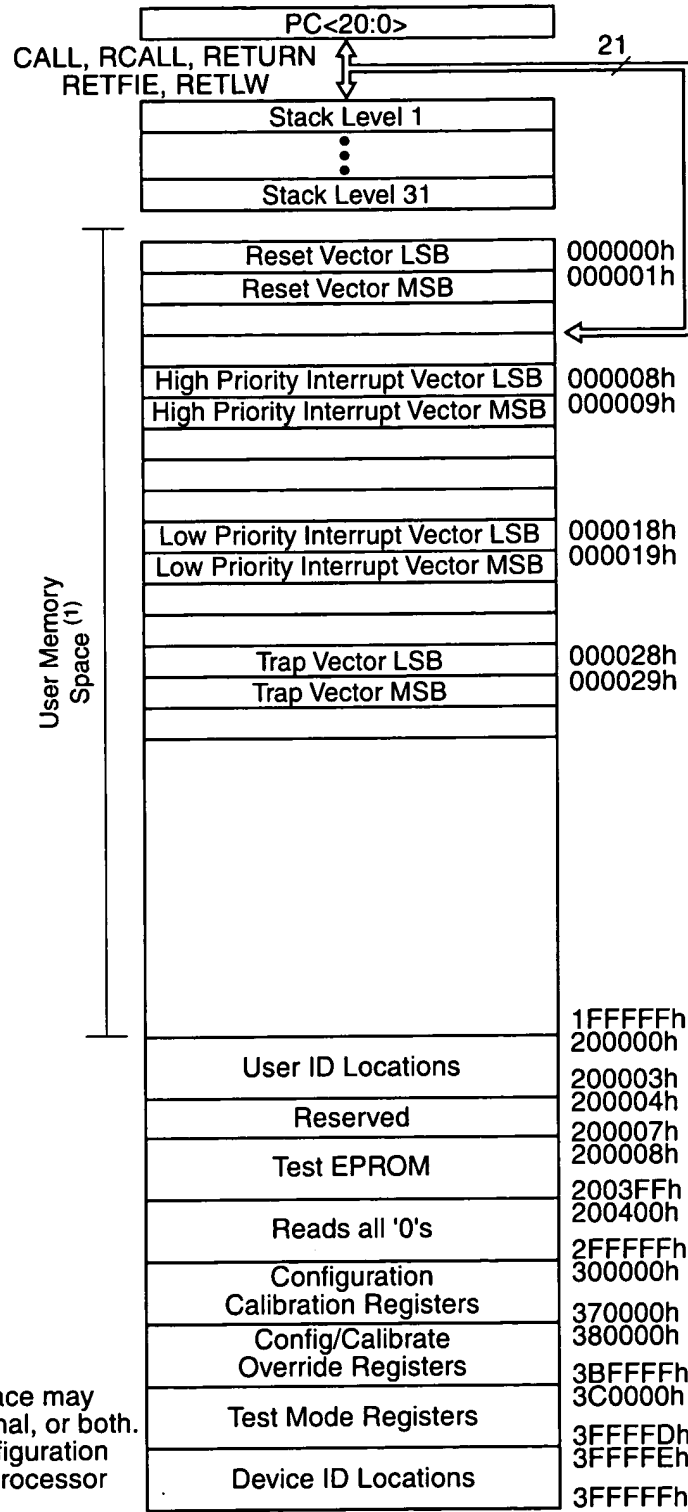
The POP instruction will perform actions similar to a return, however the PC is not loaded with the TOS value. The user must recover his data before the POP. The stack pointer is then decremented.



POP instruction

Figure 29

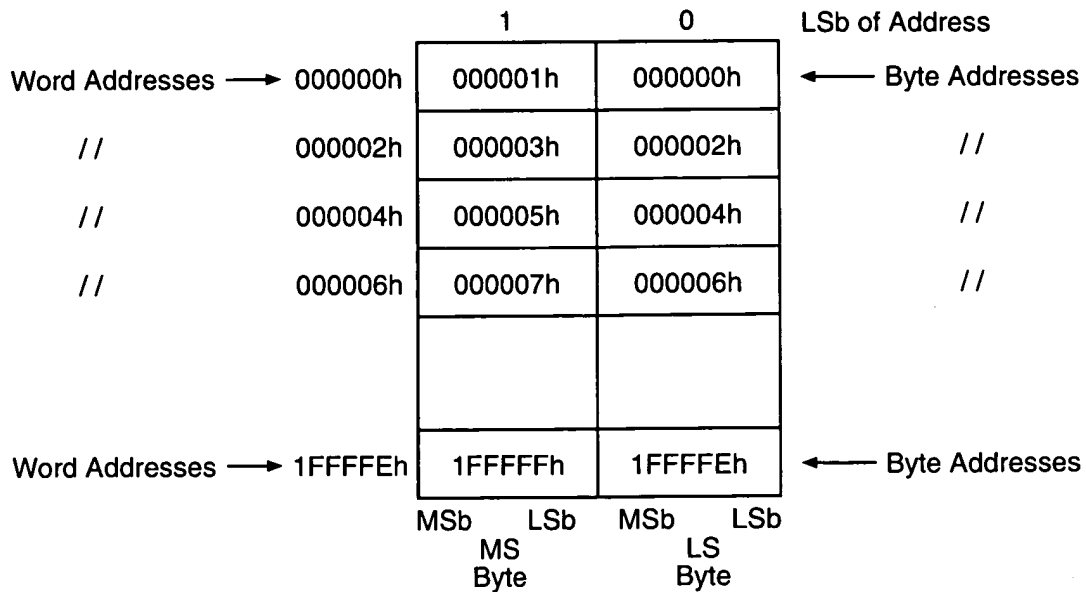
16/95



Note 1:
User memory space may be internal, external, or both. The memory configuration depends on the processor mode.

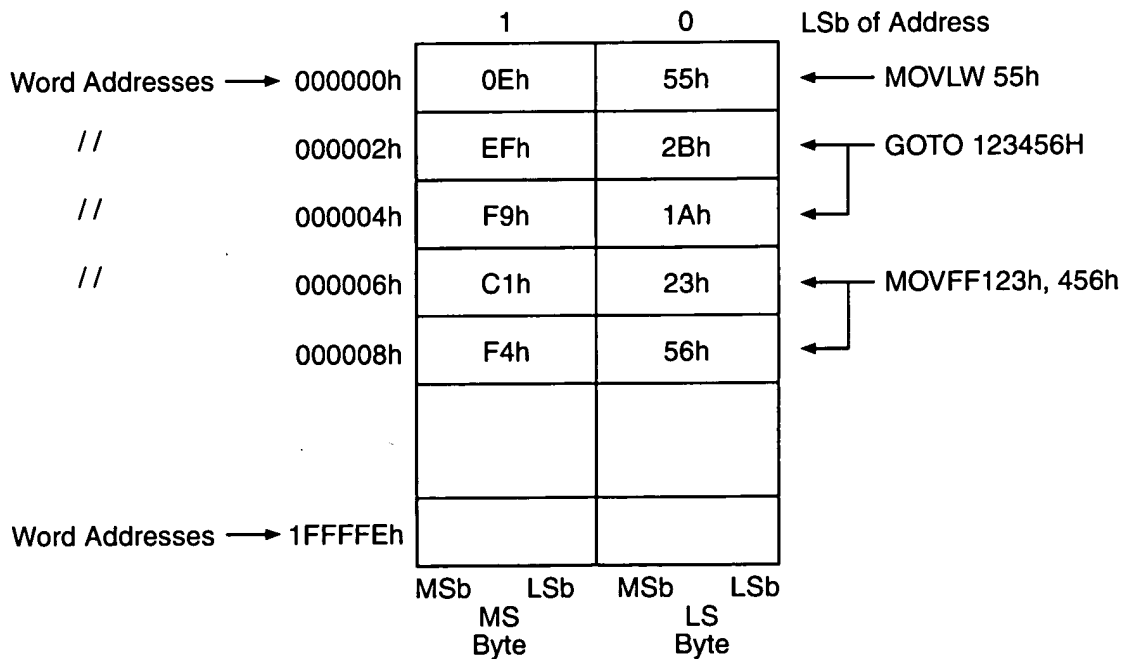
Figure 30

17/95



Memory Map in Bytes/Words

Figure 31



Instructions in Memory

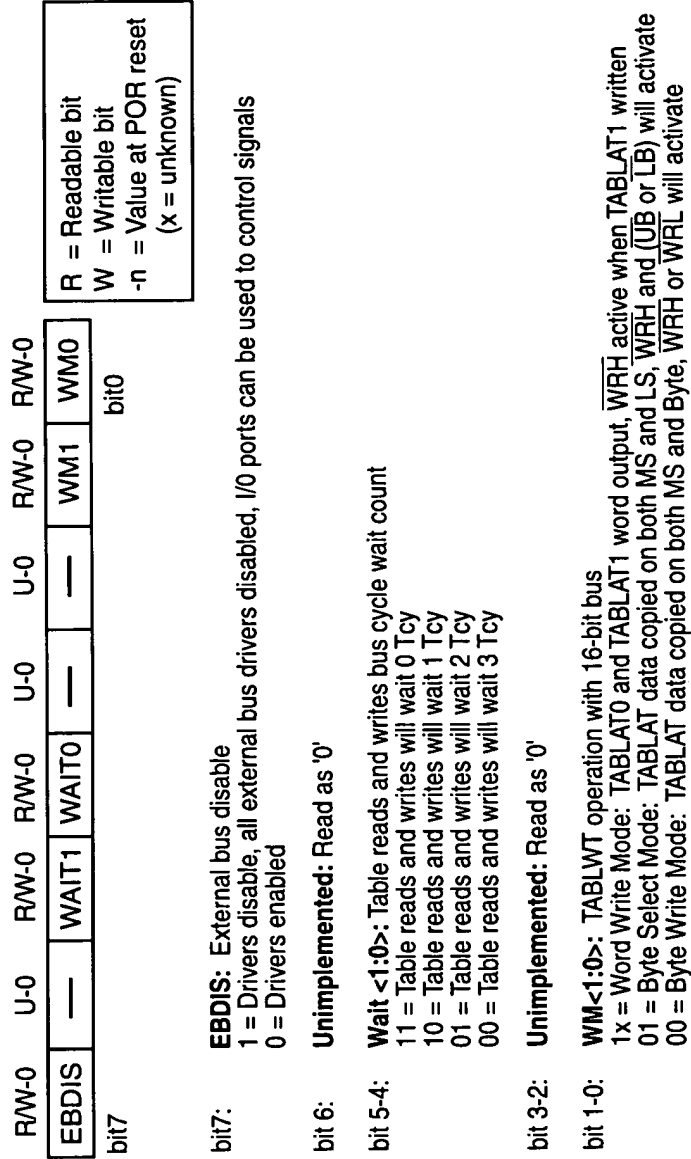
Figure 32

	Microprocessor Mode	Extended Microcontroller Mode	Microcontroller Mode
Program Space Execution	<div>000000h</div> <div>External Program Memory</div> <div>1FFFFFFh</div> <div>OFF-CHIP ON-CHIP</div>	<div>000000h</div> <div>On-chip Program Memory</div> <div>007FFFh</div> <div>008000h</div> <div>External Program Memory</div> <div>1FFFFFFh</div> <div>OFF-CHIP ON-CHIP</div>	<div>000000h</div> <div>On-chip Program Memory</div> <div>007FFFh</div> <div>008000h</div> <div>Reads '0'</div> <div>1FFFFFFh</div> <div>ON-CHIP OFF-CHIP ON-CHIP</div>
Data Space	<div>000h</div> <div>FFFh</div> <div>ON-CHIP</div>	<div>000h</div> <div>FFFh</div> <div>ON-CHIP</div>	<div>000h</div> <div>FFFh</div> <div>ON-CHIP</div>

Note 1: Example Shows 32K Bytes of Internal EPROM Program Memory

Device Memory Map in Different Modes

Figure 33



MEMCON Register

Figure 34

U-1	R/P-1	R/P-1	R/P-1	U-1	R/P-1	R/P-1	R/P-1
—	BSDIS	BADIS	WDIS	—	A19SDIS	A15DIS	A11DIS
bit7							bit0

bit 7: **Unimplemented:** Read as '0'

bit 6: **BSDIS:** Byte Select \overline{UB} , \overline{LB} disable
 0=Drivers disabled
 1=Drivers enabled

bit 5: **BADIS:** Byte Address BA0 disable
 0=Drivers disabled
 1=Drivers enabled

bit 5: **WDIS:** Write Select \overline{WRH} , \overline{WRL} disable
 0=Drivers disabled
 1=Drivers enabled

bit 3: **Unimplemented:** Read as '0'

bit 2: **A19DIS:** Disable AD19:AD16 drivers
 0=Drivers disabled
 1=Drivers enabled

bit 1: **A15DIS:** Disable AD15:AD12 drivers
 0=Drivers disabled, only if 8-bit external interface
 1=Drivers enabled

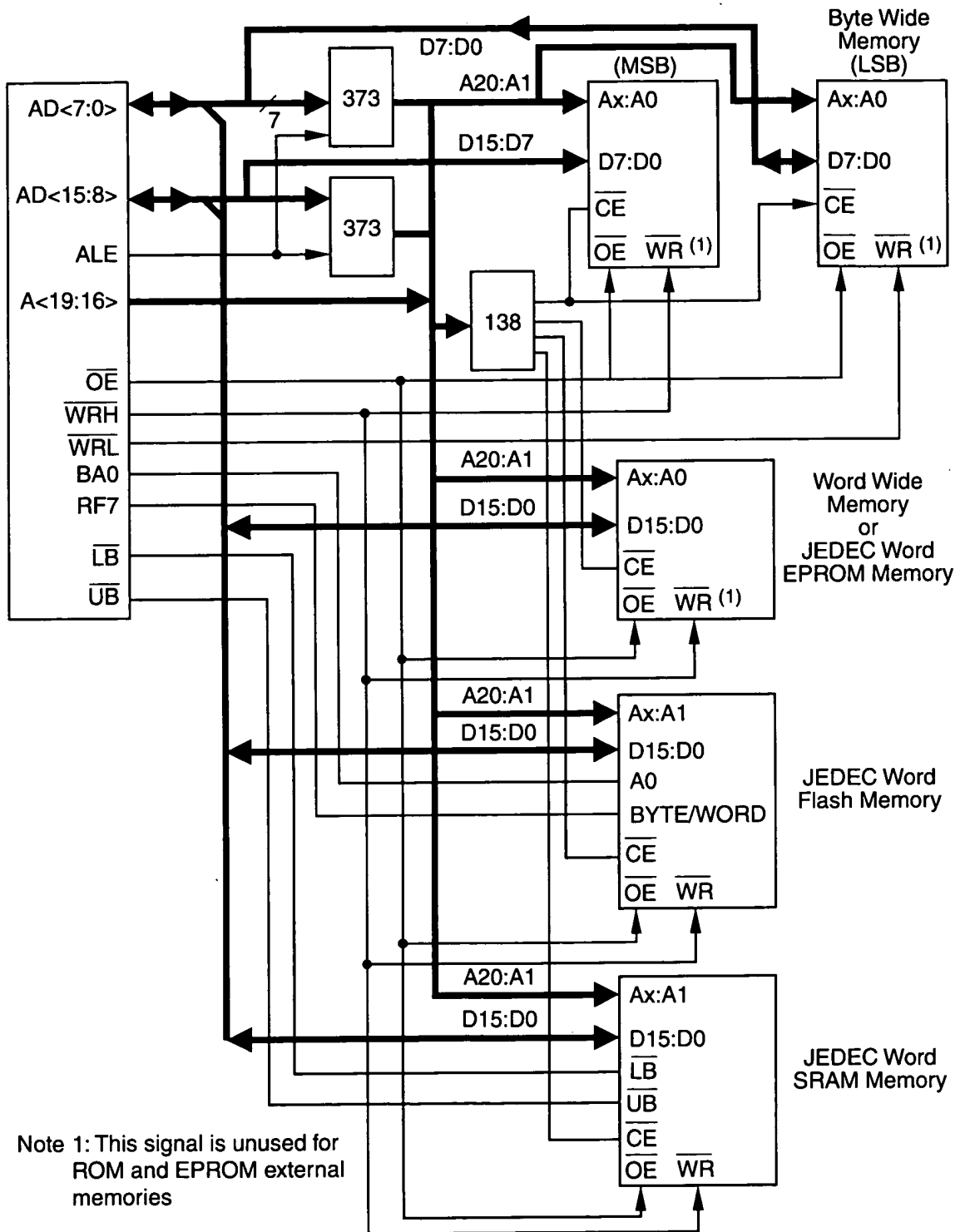
bit 0: **A11DIS:** Disable AD11:AD8 drivers
 0=Drivers disabled, only if 8-bit external interface
 1=Drivers enabled

R = Readable bit P = Programmable bit -n = UnprogrammedValue (x = unknown)

CONFIG7 Configuration Byte

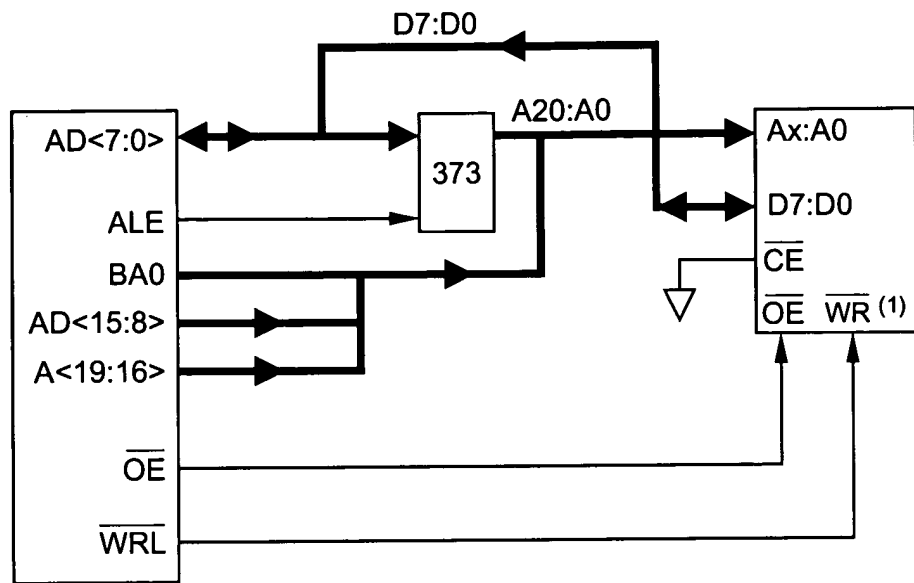
Figure 35

21/95



External Memory Connection Diagram (16-bit)

Figure 36



Note 1: This signal is unused for ROM and EPROM external memories

External Memory Connection Diagram (8-bit)

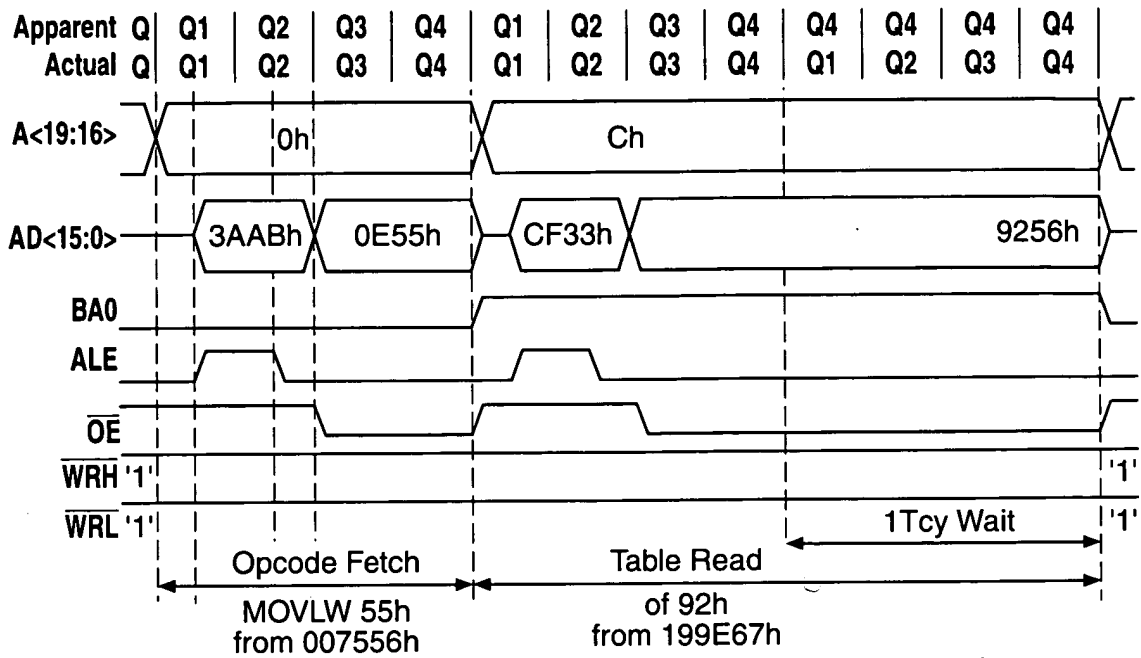
Figure 37

Name	Port	Bit	Function
RF4/BA0	PORTF	bit4	Input/Output or system bus byte address bit 0
RD0/AD0	PORTD	bit0	Input/Output or system bus address bit 0 or data bit 0
RD1/AD1	PORTD	bit1	Input/Output or system bus address bit 1 or data bit 1
RD2/AD2	PORTD	bit2	Input/Output or system bus address bit 2 or data bit 2
RD3/AD3	PORTD	bit3	Input/Output or system bus address bit 3 or data bit 3
RD4/AD4	PORTD	bit4	Input/Output or system bus address bit 4 or data bit 4
RD5/AD5	PORTD	bit5	Input/Output or system bus address bit 5 or data bit 5
RD6/AD6	PORTD	bit6	Input/Output or system bus address bit 6 or data bit 6
RD7/AD7	PORTD	bit7	Input/Output or system bus address bit 7 or data bit 7
RE0/AD8	PORTE	bit0	Input/Output or system bus address bit 8 or data bit 8
RE1/AD9	PORTE	bit1	Input/Output or system bus address bit 9 or data bit 9
RE2/AD10	PORTE	bit2	Input/Output or system bus address bit 10 or data bit 10
RE3/AD11	PORTE	bit3	Input/Output or system bus address bit 11 or data bit 11
RE4/AD12	PORTE	bit4	Input/Output or system bus address bit 12 or data bit 12
RE5/AD13	PORTE	bit5	Input/Output or system bus address bit 13 or data bit 13
RE6/AD14	PORTE	bit6	Input/Output or system bus address bit 14 or data bit 14
RE7/AD15	PORTE	bit7	Input/Output or system bus address bit 15 or data bit 15
RG0/A16	PORTG	bit0	Input/Output or system bus address bit 16
RG1/A17	PORTG	bit1	Input/Output or system bus address bit 17
RG2/A18	PORTG	bit2	Input/Output or system bus address bit 18
RG3/A19	PORTG	bit3	Input/Output or system bus address bit 19
RF0/ALE	PORTF	bit0	Input/Output or system bus Address Latch Enable (ALE) control pin
RF1/ \overline{OE}	PORTF	bit1	Input/Output or systems bus Output Enable (\overline{OE}) control pin
RF2/ \overline{WRL}	PORTF	bit2	Input/Output or system bus Write Low (\overline{WRL}) control pin
RF3/ \overline{WRH}	PORTF	bit3	Input/Output or system bus Write High (\overline{WRH}) control pin
RF5/ \overline{LB}	PORTF	bit2	Input/Output or system bus Lower Byte Enable (\overline{LB}) control pin
RF6/ \overline{UB}	PORTF	bit3	Input/Output or system bus Upper Byte Enable (\overline{UB}) control pin

Typical Port Functions

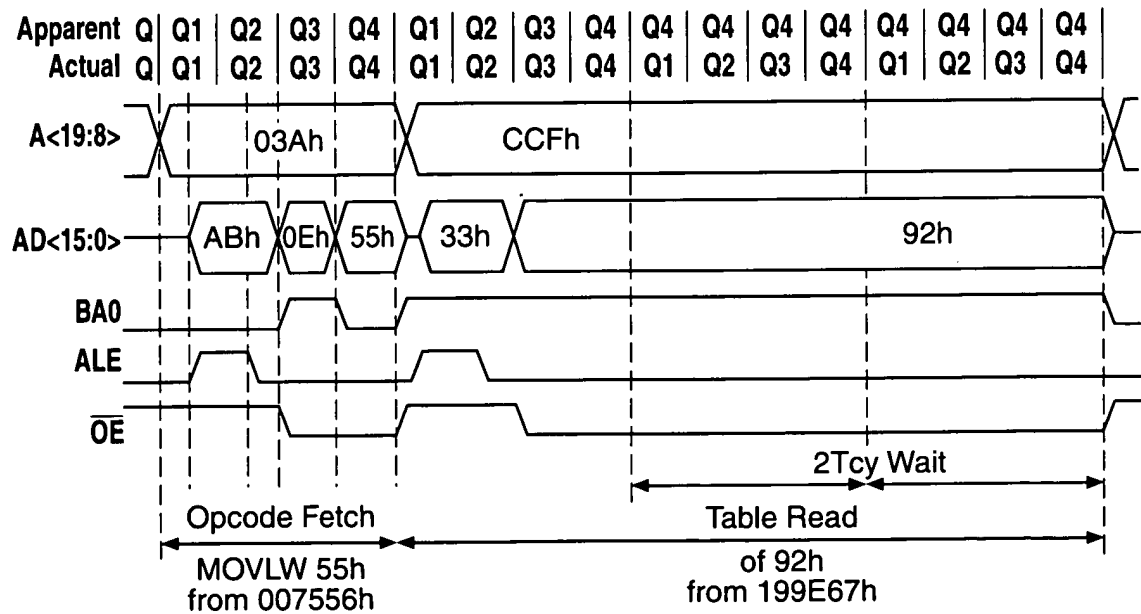
Figure 38

24/95



External Program Memory Bus Timing (16-bit Mode)

Figure 39



External Program Memory Bus Timing (8-bit Mode)

Figure 40

	A<19:16>	AD<15:8>	AD<7:0>	BA0	AL \overline{E}	OE	WRH	WRL	UB	UL
Opcode Fetch 8-bit	PC<20:17>	PC<16:9>	Q1-2:PC<8:1> Q3:INST<15:8> Q4:INST<7:0>	Q1-2:0 Q3:1 Q4:0	1	0	-	1	-	-
Opcode Fetch 16-bit	PC<20:17>	Q2:PC<20:17> Q3-4:INST<15:8>	Q1-2:PC<20:17> Q3-4:INST<7:0>	0	1	0	1	1	0	0
Table Read 8-bit	TBLPTR<20:17>	TBLPTR<16:9>	Q1-2:TBLPTR<8:1> Q3-4:DATA<7:0>	TBLPTR<0>	1	0	-	1	-	-
Table Read 16-bit	TBLPTR<20:17>	Q1-2:TBLPTR<16:9> Q3-4:DATA<15:8>	Q1-2:TBLPTR<8:1> Q3-4:DATA<7:0>	TBLPTR<0>	1	0	1	1	0	0
Table Write 8-bit	TBLPTR<20:17>	TBLPTR<16:9>	Q1-2:TBLPTR<8:1> Q3-4:TBLAT<7:0>	TBLPTR<0>	1	1	-	0	-	-
Table Write 16-bit Byte Write Mode	TBLPTR<20:17>	Q1-2:TBLPTR<16:9> Q3-4:TBLAT<7:0>	Q1-2:TBLPTR<8:1> Q3-4:TBLAT<7:0>	TBLPTR<0>	1	1	XH	XL	1	1
Table Write 16-bit Byte Select Mode	TBLPTR<20:17>	Q1-2:TBLPTR<16:9> Q3-4:TBLAT<7:0>	Q1-2:TBLPTR<8:1> Q3-4:TBLAT<7:0>	TBLPTR<0>	1	1	0	1	YH	YL
Table Write 16-bit Word Write Mode TABPTR<0>=0	TBLPTR<20:17>	Q1-2:TBLPTR<16:9> Q3-4:Hi-Z	Q1-2:TBLPTR<8:1> Q3-4:Hi-Z TBHREG<7:0>= TBLAT<7:0>	TBLPTR<0> =0	1	1	1	1	1	1
Table Write 16-bit Word Write Mode TABPTR<0>=1	TBLPTR<20:17>	Q1-2:TBLPTR<16:9> Q3-4:TBLAT<7:0>	Q1-2:TBLPTR<8:1> Q3-4:TBHREG<7:0>	TBLPTR<0> =1	1	1	0	1	0	0

TBHREG is TBLAT high byte holding register

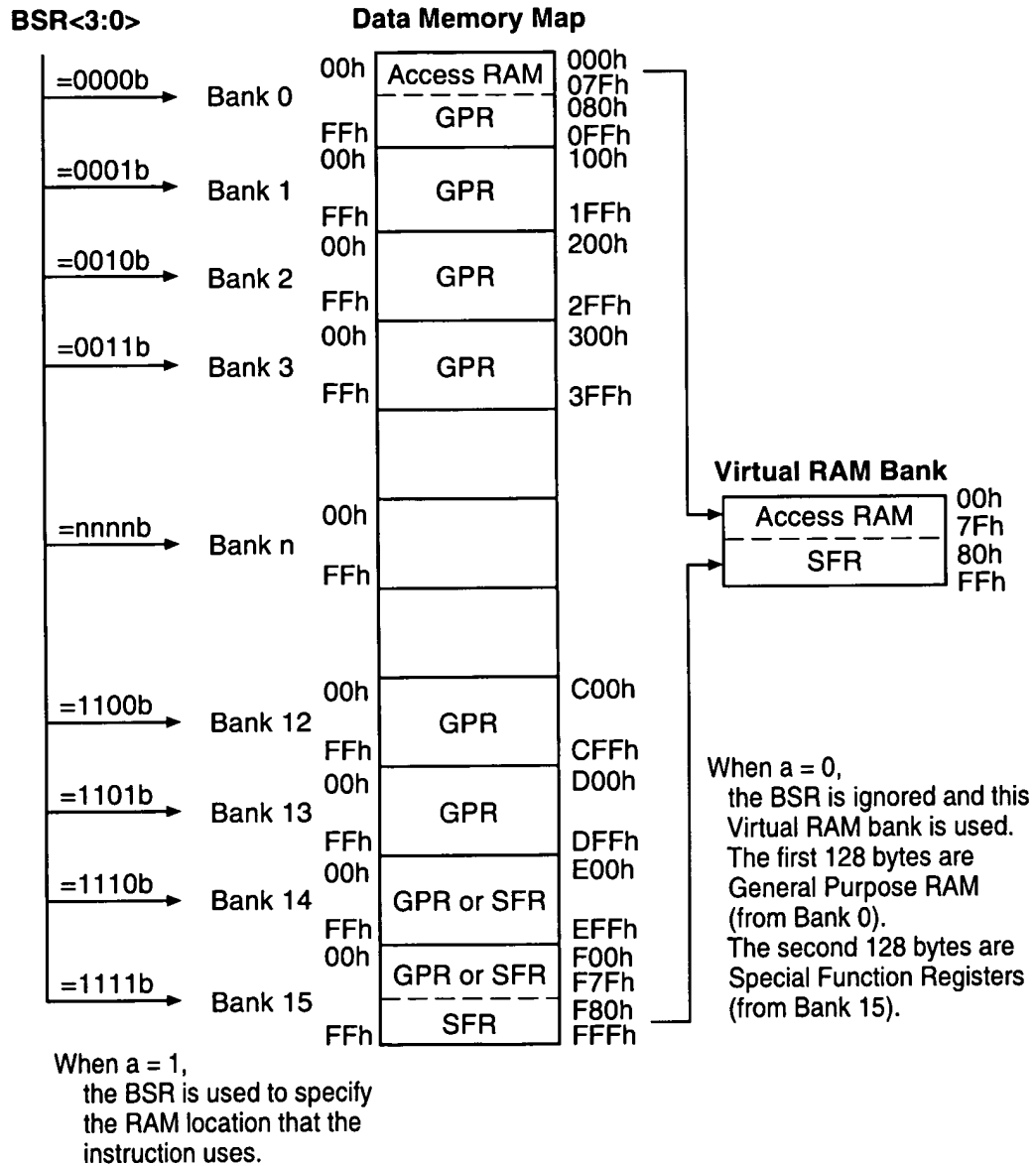
XH= WRITE signal .AND. TBLPTR<0>

XL= WRITE signal .AND. TBLPTR<0>

YH= TBLPTR<0>

YL= TBLPTR<0>

External Bus Cycle Types
Figure 41



The Data Memory Map and the Instruction 'a' bit

Figure 42

FFFh	TOSU	FDFh	INDF2	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2	FBCh	CCPR2H	F9Ch	MEMCON
FFBh	PCLATU	FDBh	PLUSW2	FBHh	CCPR2L	F9Bh	
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	DDRJ
FF9h	PCL	FD9h	FSR2L	FB9h	CCPR3H	F99h	DDRH
FF8h	TBLPTRU	FD8h	STATUS	FB8h	CCPR3L	F98h	DDRG
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	CCP3CON	F97h	DDRF
FF6h	TBLPTL	FD6h	TMR0L	FB6h	CCPR4H	F96h	DDRE
FF5h	TABLAT	FD5h	T0CON	FB5h	CCPR4L	F95h	DDRD
FF4h	PRODH	FD4h	rsvd DEBUG	FB4h	CCP4CON	F94h	DDRC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	DDRB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	DDRA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	LATJ
FF0h	INTCON3	FD0h	RCON	FB0h		F90h	LATH
FEFh	INDF0	FCFh	TMR1H	FAFh	COM1BRG	F8Fh	LATG
FEeh	POSTINC0	FCEh	TMR1L	FAEh	COM1REC	F8Eh	LATF
FEDh	POSTDEC0	FCDh	T1CON	FADh	COM1TX	F8Dh	LATE
FECh	PREINC0	FCCh	TMR2	FACH	COM1STA	F8Ch	LATD
FEBh	PLUSW0	FCBh	PR2	FABh	COM1CON	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	COM2BRG	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	COM2REC	F89h	LATA
FE8h	W	FC8h	SSPAD	FA8h	COM2TX	F88h	PORTJ
FE7h	INDF1	FC7h	SSPSTAT	FA7h	COM2STA	F87h	PORTH
FE6h	POSTINC1	FC6h	SSPCON1	FA6h	COM2CON	F86h	PORTG
FE5h	POSTDEC1	FC5h	SSPCON2	FA5h	IPR3	F85h	PORTF
FE4h	PREINC1	FC4h	ADRESH	FA4h	PIR3	F84h	PORTE
FE3h	PLUSW1	FC3h	ADRESL	FA3h	PIE3	F83h	PORTD
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA

Special Function Register Map

Figure 43

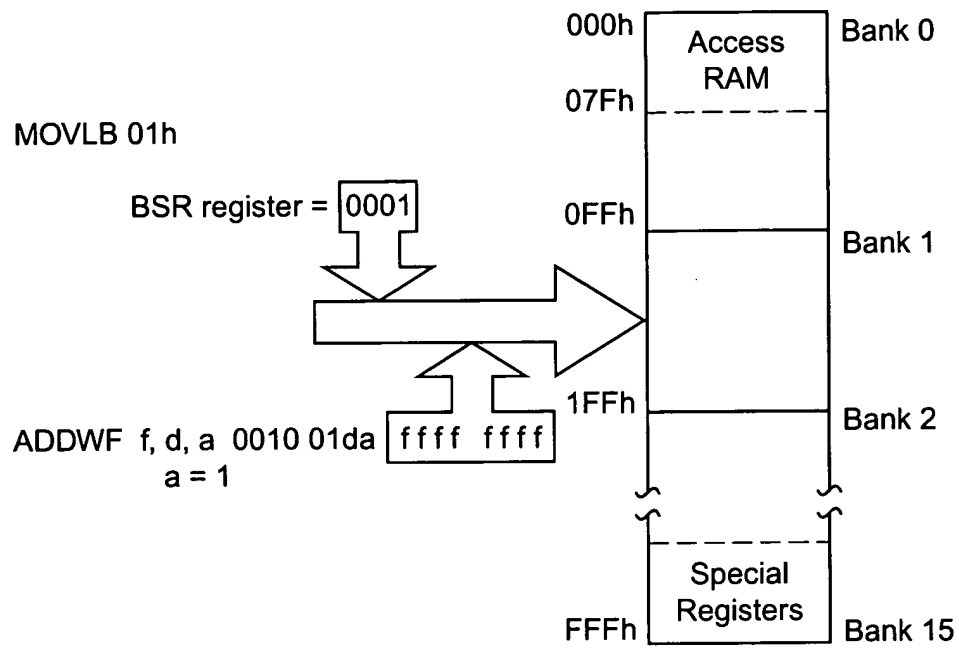
Filename	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other resets (note 3)	
FFh	TOSU	-	-	-	Top-of-Stack upper Byte (TOS<20:16>)				-- 0 0000	-- 0 0000	
FEh	TOSH	Top-of-Stack High Byte (TOS<15:8>)							0000 0000	0000 0000	
FDh	TOSL	Top-of-Stack Low Byte (TOS<7:0>)							0000 0000	0000 0000	
FC	STKPTR	STKOVF	STKUNF	-	Return Stack Pointer					00 - 0 0000	00 - 0 0000
FBh	PCLATU	-	-	Holding Register for PC<21:16>						-- 00 0000	-- 00 0000
FAh	PCLATH	Holding Register for PC<15:8>							0000 0000	0000 0000	
F9h	PCL	PC Low Byte (PC<7:0>)							0000 0000	0000 0000	
F8h	TBLPTRU	-	-	Program Memory Table Pointer Upper Byte (TBLPTR<21:16>)						-- 00 0000	-- 00 0000
F7h	TBLPTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)							0000 0000	0000 0000	
F6h	TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)							0000 0000	0000 0000	
F5h	TABLAT	Program Memory Table Latch							0000 0000	0000 0000	
F4h	PRODH	Product Register High Byte							xxxx xxxx	uuuu uuuu	
F3h	PRODL	Product Register Low Byte							xxxx xxxx	uuuu uuuu	
F2h	INTCON	GIE/GIEH	PEIE/GIEL	T0IE	INT0E	RBIE	T0IF	INT0F	RBIF	0000 000x	0000 000x
F1h	INTCON2	RBPJ	INTEDG0	INTEDG1	INTEDG2	INTEDG3	T0IP	INT3P	RPIP	1111 1111	1111 1111
F0h	INTCON3	INT2P	INT1P	INT3E	INT2E	INT1E	INT3F	INT2F	INT1F	1100 0000	1100 0000
EFh	INDF0	Uses contents of FSR0 to address data memory - value of FSR0 not changed (not a physical register)							n/a	n/a	
EEh	POSTINC0	Uses contents of FSR0 to address data memory - value of FSR0 post-increment (not a physical register)							n/a	n/a	
EDh	POSTDEC0	Uses contents of FSR0 to address data memory - value of FSR0 post-decrement (not a physical register)							n/a	n/a	
ECh	PREINC0	Uses contents of FSR0 to address data memory - value of FSR0 pre-incremented (not a physical register)							n/a	n/a	
EBh	PLUSW0	Uses contents of FSR0 to address data memory - value of FSR0 offset by W (not a physical register)							n/a	n/a	
EAh	FSR0H	-	-	-	-	Indirect Data Memory Address Pointer 0 High				---- xxxx	---- uuuu
E9h	FSR0L	Indirect Data Memory Address Pointer 0 Low Byte							xxxx xxxx	uuuu uuuu	
E8h	W	Working Register							xxx xxxx	uuuu uuuu	
E7h	INDF1	Uses contents of FSR1 to address data memory - value of FSR1 not changed (not a physical register)							n/a	n/a	
E6h	POSTINC1	Uses contents of FSR1 to address data memory - value of FSR1 post-increment (not a physical register)							n/a	n/a	
E5h	POSTDEC1	Uses contents of FSR1 to address data memory - value of FSR1 post-decrement (not a physical register)							n/a	n/a	
E4h	PREINC1	Uses contents of FSR1 to address data memory - value of FSR1 pre-incremented (not a physical register)							n/a	n/a	
E3h	PLUSW1	Uses contents of FSR1 to address data memory - value of FSR1 offset by W (not a physical register)							n/a	n/a	
E2h	FSR1H	-	-	-	-	Indirect Data Memory Address Pointer 1 High				---- xxxx	---- uuuu
E1h	FSR1L	Indirect Data Memory Address Pointer 1 Low Byte							xxxx xxxx	uuuu uuuu	
E0h	BSR	-	-	-	-	Bank Select Register				---- 0000	---- 0000
DFh	INDF2	Uses contents of FSR2 to address data memory - value of FSR2 not changed (not a physical register)							n/a	n/a	
DEh	POSTINC2	Uses contents of FSR2 to address data memory - value of FSR2 post-increment (not a physical register)							n/a	n/a	
DDh	POSTDEC2	Uses contents of FSR2 to address data memory - value of FSR2 post-decrement (not a physical register)							n/a	n/a	
DCh	PREINC2	Uses contents of FSR2 to address data memory - value of FSR2 pre-incremented (not a physical register)							n/a	n/a	
DBh	PLUSW2	Uses contents of FSR2 to address data memory - value of FSR2 offset by W (not a physical register)							n/a	n/a	
DAh	FSR2H	-	-	-	-	Indirect Data Memory Address Pointer 2 High				---- xxxx	---- uuuu
D9h	FSR2L	Indirect Data Memory Address Pointer 2 Low Byte							xxxx xxxx	uuuu uuuu	
D8h	STATUS	-	-	-	N	OV	Z	DC	C	--- x xxxx	--- u uuuu

Core Special Function Register Summary

Figure 44

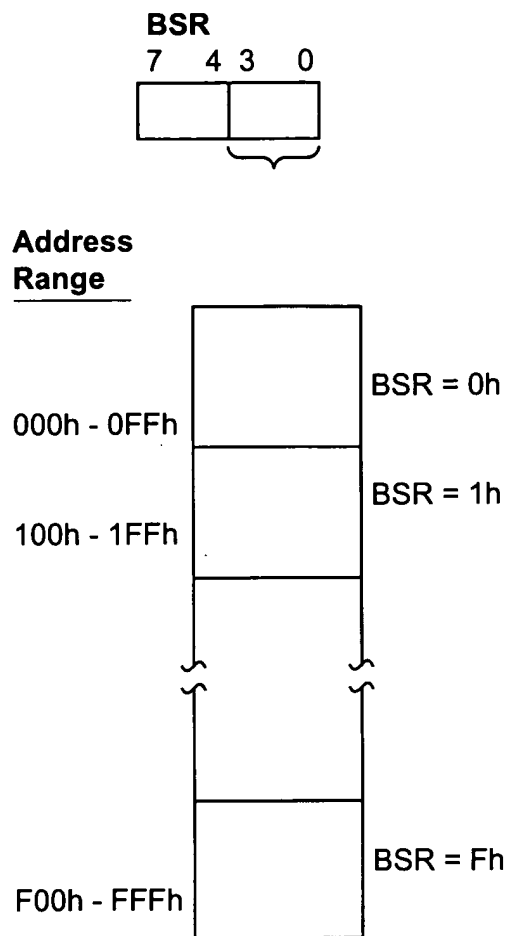
Filename		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other resets (note 3)
D3h	OSCCON	-	-	-	-	-	-	-	SCS	---0	---0
D2h	LVDCON	-	-	BGST	LV DEN	LVV3	LVV2	LVV1	LVV0	-00 0101	-00 0101
D1h	WDTCN	-	-	-	-	-	-	-	SWDTE	---0	---0
D0h	RCON	IPE	LWRT	-	RI	TO	PD	POR	BOR	00-1 11qq	00-q qqqu
A5h	IPR3	-	-	-	-	-	-	-	-	----	----
A4h	PIR3	-	-	-	-	-	-	-	-	----	----
A3h	PIE3	-	-	-	-	-	-	-	-	----	----
A2h	IPR2	-	-	-	-	BCLIP	LVDIP	TMR3IP	CCP2IP	--- 1111	--- 1111
A1h	PIR2	-	-	-	-	BCLIF	LVDIF	TMR3IF	CCP2IF	--- 0000	--- 0000
A0h	PIE2	-	-	-	-	BCLIE	LVDIE	TMR3IE	CCP2IE	---- 0000	---- 0000
9Fh	IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
9Eh	PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
9Dh	PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
9Ch	MEMCON	EBDIS	-	WAIT1	WAIT0	-	-	WM1	WM0	0-00 -00	0-00 -00

Figure 45

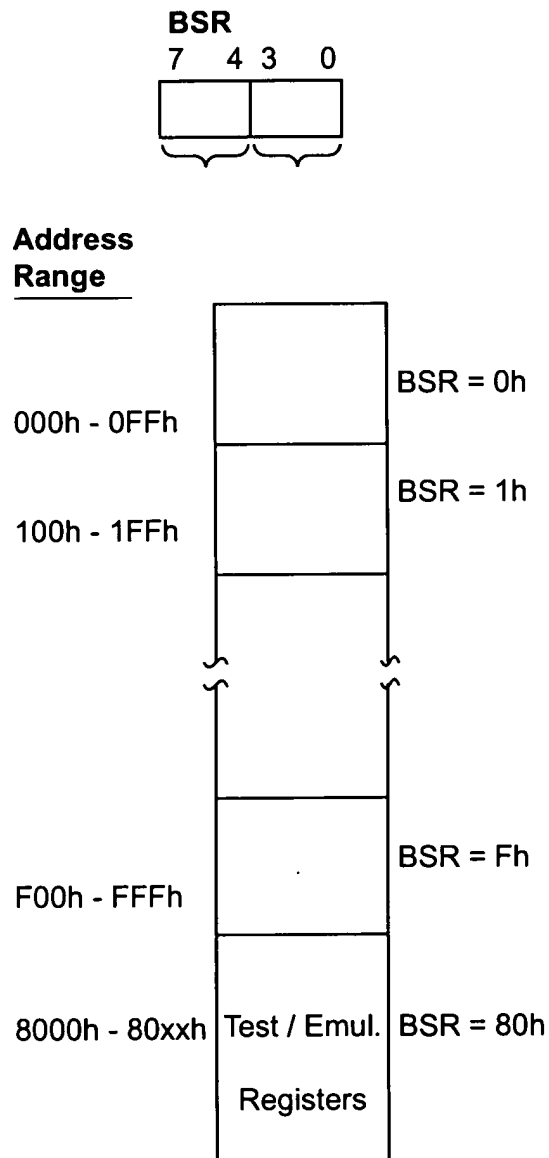


Direct Short Addressing Mode

Figure 46



BSR Operation
Figure 47



BSR Operation During Emulation/Test Modes
Figure 48

32/95

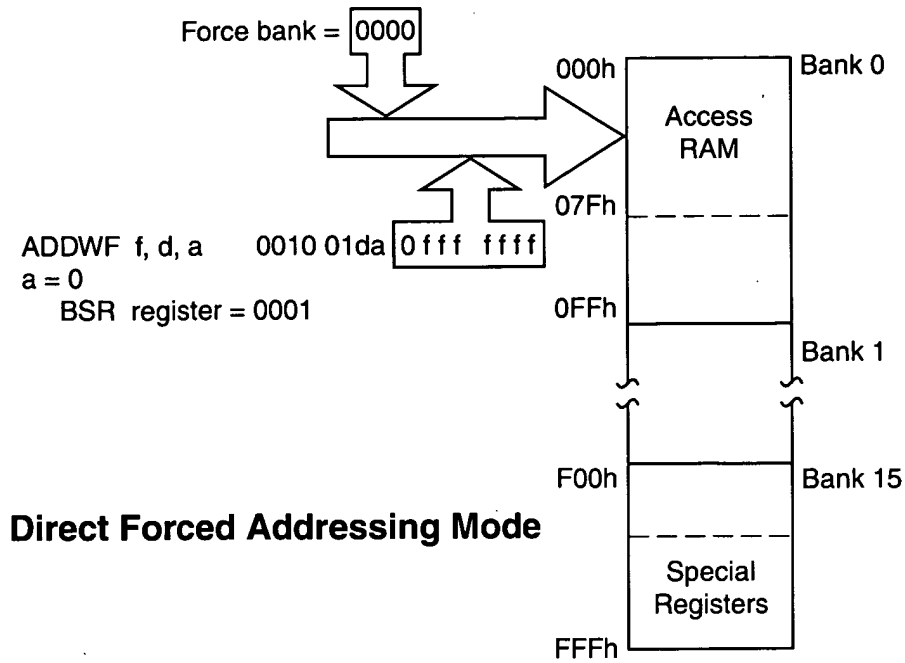


Figure 49

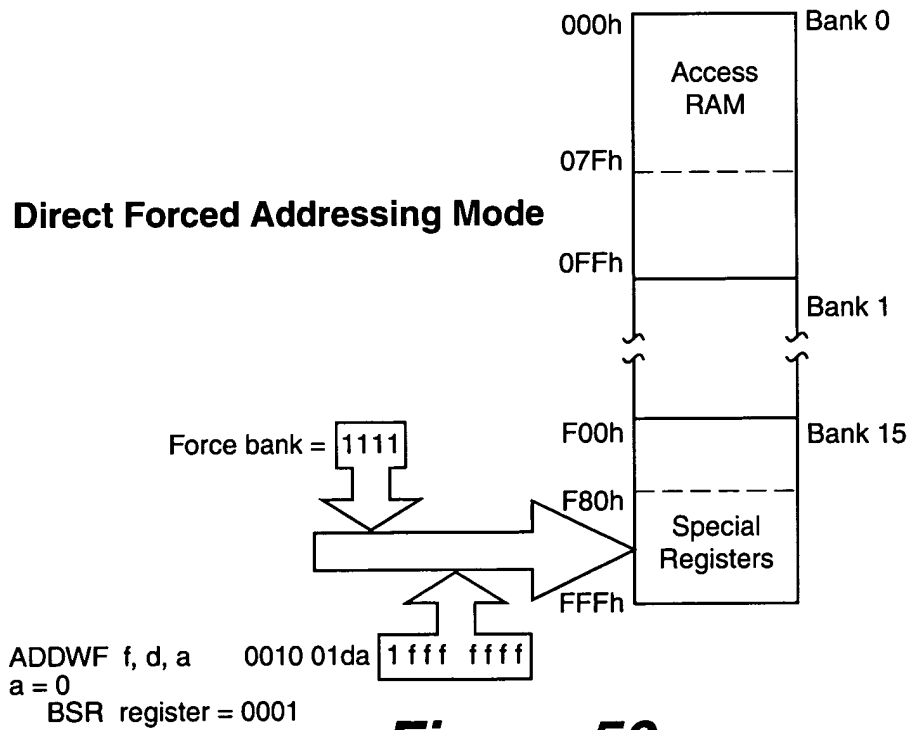
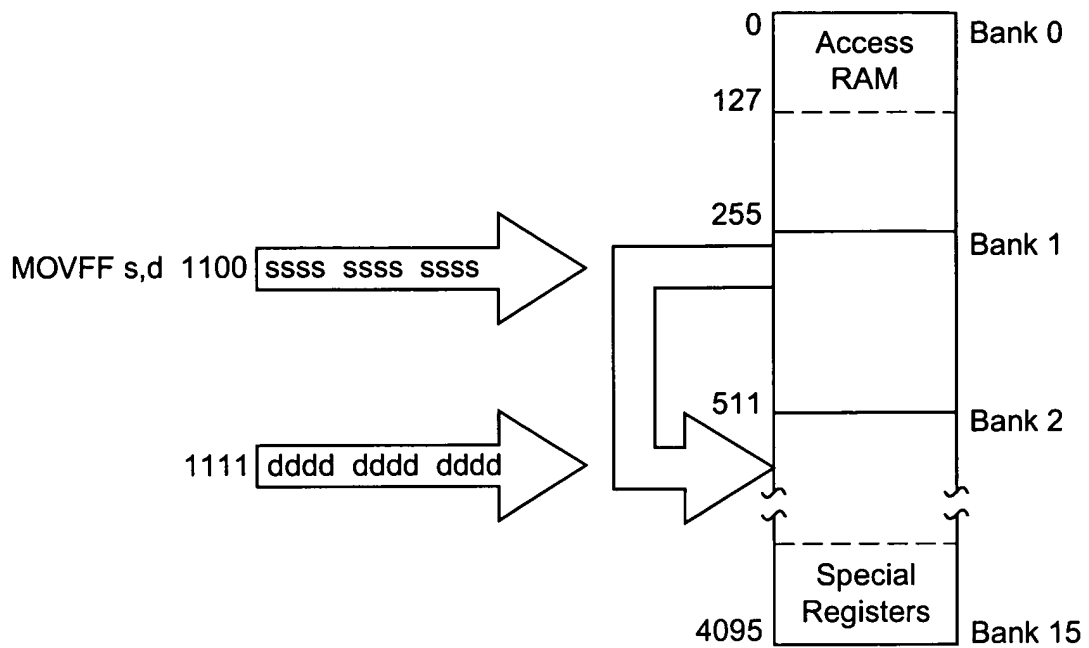
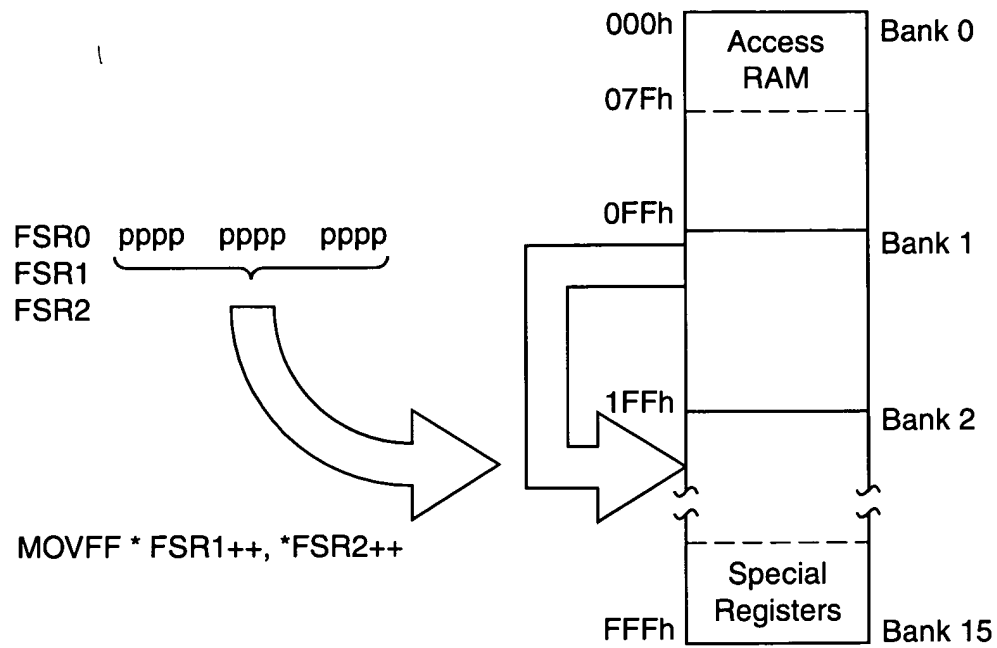


Figure 50



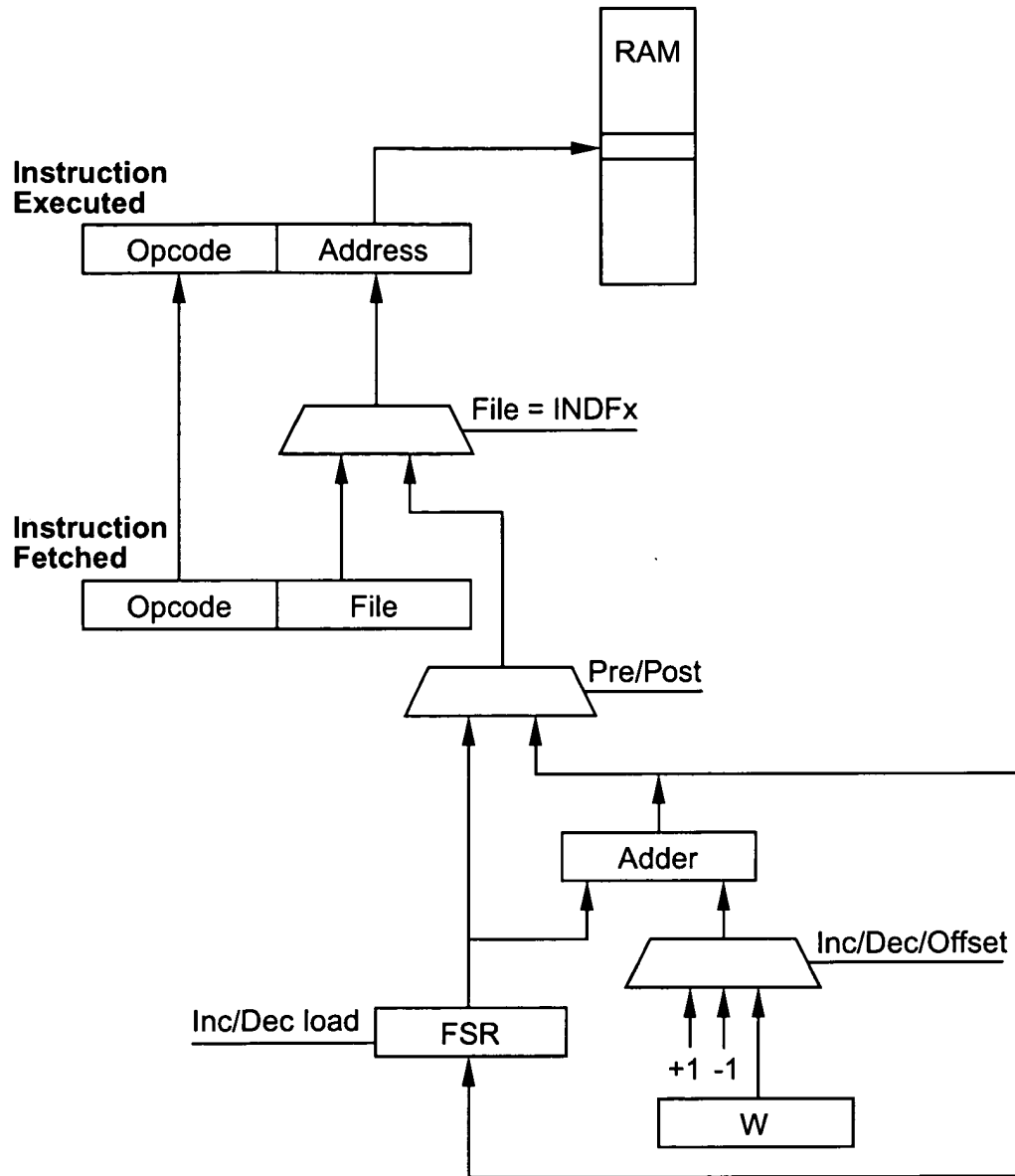
Direct Long Addressing Mode

Figure 51



Indirect Addressing Mode

Figure 52



Indirect Addressing

Figure 53

Field	Description
f fs fd	Register file address (00h to FFh) in current or virtual bank, except MOVFF (000h to FFFh)
a	Virtual bank select 0 = override BSR and force virtual bank 1 = do not override BSR Default is a = '1'
s	Fast call/return select 0 = do not update into/from shadow registers 1 = certain registers loaded into/from shadow registers Default is s = '0'
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
n	2's complement number for relative branch instructions
x	Don't care location (= '0' or '1') The assembler will generate code with x = '0'. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select 0 = store result in W 1 = store result in file register f Default is d = '1'
u	Unused, encoded as '0'
label	Label name
C, DC, Z, OV, N	ALU status bits Carry, Digit Carry, Zero, Overflow, Negative
GIE/ GIEH	Global Interrupt Enable bit (INTCON<7>)
PEIE/ GIEL	Low Priority Interrupt Enable bit (INTCON<6>)
TBLPTRU TBLPTRH TBLPTRL	Table Pointer (21-bit)
TABLAT	Table Latch (8-bit)
PRODL	Product of Multiply low byte
PRODH	Product of Multiply high byte
TOSU TOSH TOSL	Top of Stack

Field	Description
PCU PCH PCL	Program Counter
BSR	Bank Select Register
WDT	Watchdog Timer Counter
TO	Time-out bit
PD	Power-down bit
dest	Destination either the W register or the specified register file location
*	No Change to TBLPTR
*+	Post-Increment TBLPTR
*-	Post-Decrement TBLPTR
++	Pre-Increment TBLPTR
[]	Options
()	Contents
→	Assigned to
<>	Register bit field

Opcode Field Descriptions

Figure 54

Field	Description
*FSRn	Selects INDFn Register
*FSRn++	Selects POSTINCn Register
*FSRn--	Selects POSTDECn Register
*(++FSRn)	Selects PREINCn Register
*(FSRn+W)	Selects PLUSWn Register

Indirect Addressing Symbols

Figure 55

Figures 54 and 55 list the symbols recognized by the MPASM assembler.

Note 1: Any unused opcode is Reserved.
Use of any reserved opcode may cause unexpected operation.

All instruction examples use the following format to represent a hexadecimal number:

0xnn

where 0x signifies a hexadecimal digit.

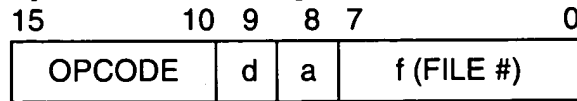
To represent a binary number:

nnnnnnnnb

where b signifies a binary string.

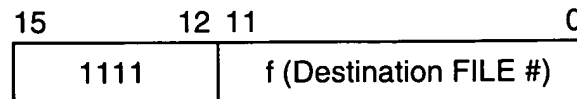
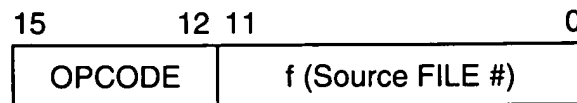
37/95

Byte-oriented file register operations



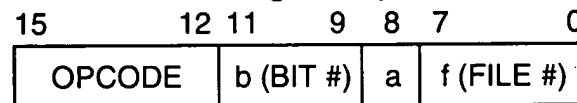
d = 0 for destination W
 d = 1 for destination f
 a = 0 for force Virtual bank
 a = 1 for BSR to select bank
 f = 8-bit file register address

Byte to Byte move operations (2-word)



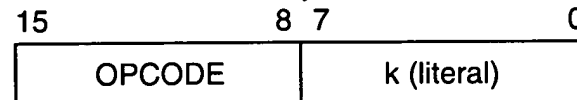
f = 12-bit file register address

Bit-oriented file register operations



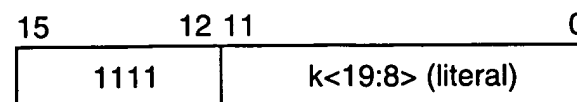
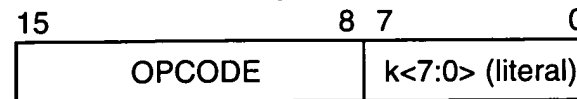
b = 3-bit address
 a = 0 for force Virtual bank
 a = 1 for BSR to select bank
 f = 8-bit file register address

Literal and control operations



k = 8-bit immediate value

CALL and GOTO operations



k = 20-bit immediate value

General Format for Instructions

Figure 56

Mnemonic, Operands	Description	Cycles	16-bit Opcode		Status Affected	Notes
			MSb	LSb		
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF f,d,a	ADD W to f	1	0010	01da ffff ffff	C,DC,N,OV,Z	3
ADDWFC f,d,a	ADD W and Carry bit to f	1	0010	00da ffff ffff	C,D,C,N,OV,Z	3
ANDWF f,d,a	AND W with f	1	0001	01da ffff ffff	N,Z	3
CLRF f,a	Clear f	1	0110	101a ffff ffff	Z	3
COMF f,d,a	Complement f	1	0001	11da ffff ffff	N,Z	3
CPFSEQ f,a	Compare f with W, skip if f = W	1 (2)	0110	001a ffff ffff	None	3,5,7,8
CPFSGT f,a	Compare f with W, skip if f > W	1 (2)	0110	010a ffff ffff	None	2,3,5,7,8
CPFSLT f,a	Compare f with W, skip if f < W	1 (2)	0110	000a ffff ffff	None	2,3,5,7,8
DECF f,d,a	Decrement f	1	0000	01da ffff ffff	C,DC,N,OV,Z	3
DECFSZ f,d,a	Decrement f, skip if 0	1 (2)	0010	11da ffff ffff	None	3,5,7,8
DCFSNZ f,d,a	Decrement f, skip if not 0	1 (2)	0100	11da ffff ffff	None	3,5,7,8
INCF f,d,a	Increment f	1	0010	10da ffff ffff	C,DC,N,OV,Z	3
INCSZ f,d,a	Increment f, skip if 0	1 (2)	0011	11da ffff ffff	None	3,5,7,8
INFSNZ f,d,a	Increment f, skip if not 0	1 (2)	0100	10da ffff ffff	None	3,5,7,8
IORWF f,d,a	Inclusive OR W with f	1	0001	00da ffff ffff	N,Z	3
MOVF f,d,a	Move f	1	0101	00da ffff ffff	N,Z	3
MOVFF f _s ,f _d	Move f _s (1st word) to f _d (2nd word)	2	1100 ffff ffff ffff 1111 ffff ffff ffff		None	6
MOVWF f,a	Move W to f	1	0110	111a ffff ffff	None	3
MULWF f,a	Multiply W with f	1	0000	001a ffff ffff	None	3
NEGF f,a	Negate f	1	0110	110a ffff ffff	C,DC,N,OV,Z	1,3
NOP ---	No Operation	1	0000	0000 0000 0000	None	
NOP ---	No Operation (2nd Word)	1	1111	xxxx xxxx xxxx	None	
RLCF f,d,a	Rotate left f though Carry	1	0011	01da ffff ffff	C,N,Z	3
RLNCF f,d,a	Rotate left f (no carry)	1	0100	01da ffff ffff	N,Z	3
RRCF f,d,a	Rotate right f through Carry	1	0011	00da ffff ffff	C,N,Z	3
RRNCF f,d,a	Rotate right f (no carry)	1	0100	00da ffff ffff	N,Z	3
SETF f,a	Set f	1	0110	100a ffff ffff	None	3

Legend: Refer to Table 3-6 for opcode field descriptions.

Note 1: 2's Complement method.

2: Unsigned arithmetic.

3: If a = '0', the Bank Select Register (BSR) will be overridden and Virtual bank is selected; If a = '1', the BSR is used.

4: Multiple cycle instruction for EPROM programming when table pointer selects internal EPROM. The instruction is terminated by an interrupt event. Writing to external program memory is a two-cycle instruction.

5: Two-cycle instruction when condition is true, else single cycle instruction.

6: Two-cycle instruction except for MOVFF to PCL (program counter low byte) in which case it takes 3 cycles.

7: A "skip" means that the instruction fetched during execution of the current instruction is not executed, instead an NOP is executed.

8: When a "skip" instruction executes a skip and is followed by a 2-word instruction, 3 cycles will be executed.

9: If s = '1', certain registers will be loaded from/into shadow registers. If s = '0' no update occurs.

Instruction Set Summary

Figure 57

39/95

Mnemonic, Operands	Description	Cycles	16-bit Opcode		Status Affected	Notes
			MSb	LSb		
SUBFWB f,d,a	Subtract f from W with Borrow	1	0101 01da	ffff ffff	C,DC,N,OV,Z	1,3
SUBWF f,d,a	Subtract W from f	1	0101 11da	ffff ffff	C,DC,N,OV,Z	1,3
SUBWFB f,d,a	Subtract W from f with Borrow	1	0101 10da	ffff ffff	C,DC,N,OV,Z	1,3
SWAPF f,d,a	Swap f	1	0011 10da	ffff ffff	None	3
TSTFSZ f,a	Test f, skip if 0	1 (2)	0110 011a	ffff ffff	None	3,5,7,8
XORWF f,d,a	Exclusive OR W with f	1	0001 10da	ffff ffff	N,Z	3
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF f,b,a	Bit Clear f	1	1001 bbba	ffff ffff	None	3
BSF f,b,a	Bit Set f	1	1000 bbba	ffff ffff	None	3
BTFSZ f,b,a	Bit test f, skip if clear	1 (2)	1011 bbba	ffff ffff	None	3,5,7,8
BTFSZ f,b,a	Bit test f, skip if set	1 (2)	1010 bbba	ffff ffff	None	3,5,7,8
BTG f,b,a	Bit Toggle f	1	0111 bbba	ffff ffff	None	3
LITERAL AND CONTROL OPERATIONS						
ADDLW k	ADD literal to W	1	0000 1111	kkkk kkkk	C,DC,N,OV,Z	
ANDLW k	AND literal with W	1	0000 1011	kkkk kkkk	N,Z	
BC n	Branch if Carry	1 (2)	1110 0010	nnnn nnnn	None	
BN n	Branch if Negative	1 (2)	1110 0110	nnnn nnnn	None	
BNC n	Branch if Not Carry	1 (2)	1110 0011	nnnn nnnn	None	
BNN n	Branch if Not Negative	1 (2)	1110 0111	nnnn nnnn	None	
BNV n	Branch if Not Overflow	1 (2)	1110 0101	nnnn nnnn	None	
BNZ n	Branch if Not Zero	1 (2)	1110 0001	nnnn nnnn	None	
BRA n	Unconditional branch	2	1101 0nnn	nnnn nnnn	None	
BV n	Branch if Overflow	1 (2)	1110 0100	nnnn nnnn	None	
BZ n	Branch if Zero	1 (2)	1110 0000	nnnn nnnn	None	
CALL k,s	Subroutine Call (1st word) (2nd word)	2	1110 110s	kkkk kkkk 1111 kkkk kkkk kkkk	None	9
CLRWDT -	Clear Watchdog Timer	1	0000 0000	0000 0100	TO,PD	
DAW -	Decimal Adjust W Register	1	0000 0000	0000 0111	C	
GOTO k	Unconditional Branch (1st word) (2nd word)	2	1110 1111	kkkk kkkk 1111 kkkk kkkk kkkk	None	
HALT -	Halt processor	1	0000 0000	0000 0001	None	
IORLW k	Inclusive OR literal with W	1	0000 1001	kkkk kkkk	N,Z	

Legend: Refer to Table 3-6 for opcode field descriptions.

Note 1: 2's Complement method.

2: Unsigned arithmetic.

3: If a = '0', the Bank Select Register (BSR) will be overridden and Virtual bank is selected: If a = '1', the BSR is used.

4: Multiple cycle instruction for EPROM programming when table pointer selects internal EPROM. The instruction is terminated by an interrupt event. Writing to external program memory is a two-cycle instruction.

5: Two-cycle instruction when condition is true, else single cycle instruction.

6: Two-cycle instruction except for MOVFF to PCL (program counter low byte) in which case it takes 3 cycles.

7: A "skip" means that the instruction fetched during execution of the current instruction is not executed, instead an NOP is executed.

8: When a "skip" instruction executes a skip and is followed by a 2-word instruction, 3 cycles will be executed.

9: If s = '1', certain registers will be loaded from/into shadow registers. If s = '0' no update occurs.

Instruction Set Summary (Continued)

Figure 58

Mnemonic, Operands	Description	Cycles	16-bit Opcode		Status Affected	Notes
			MSb	LSb		
LFSR f,k	Move Literal to FSR (second word)	2	1110 1110 00ff kkkk 1111 0000 kkkk kkkk		None	
MOVLB k	Move literal to low nibble in BSR	1	0000 0001 0000 kkkk		None	
MOVLW k	Move literal to W	1	0000 1110 kkkk kkkk		None	
MULLW k	Multiply literal with W	1	0000 1101 kkkk kkkk		None	
POP ---	Pop Top of return stack (TOS)	1	0000 0000 0000 0110		None	
PUSH ---	Push Top of return stack (TOS)	1	0000 0000 0000 0101		None	
RCALL n	Unconditional subroutine branch	2	1101 1nnn nnnn nnnn		None	
RESET ---	Generate a Reset (same as MCLR reset)	1	0000 0000 1111 1111		All - Reset	
RETFIE s	Return from interrupt (and enable interrupts)	2	0000 0000 0001 000s		GIEH,GIEL All if s=1	9
RETLW k	Return literal to W	2	0000 1100 kkkk kkkk		None	
RETURN s	Return from Subroutine	2	0000 0000 0001 001s		None if s=0 All if s=1	9
SLEEP ---	Enter SLEEP Mode	1	0000 0000 0000 0011		TO, PD	
SUBLW k	Subtract W from literal	1	0000 1000 kkkk kkkk		N,OV,C,DC,Z	
TBLRD* ---	Table Read (no change to TBLPTR)	2	0000 0000 0000 1000		None	
TBLRD*+ ---	Table Read (post-increment TBLPTR)	2	0000 0000 0000 1001		None	
TBLRD* ---	Table Read (post-decrement TBLPTR)	2	0000 0000 0000 1010		None	
TBLRD*+ ---	Table Read (pre-increment TBLPTR)	2	0000 0000 0000 1011		None	
TBLWT* ---	Table Write (no change to TBLPTR)	2	0000 0000 0000 1100		None	4
TBLWT*+ ---	Table Write (post-increment TBLPTR)	2	0000 0000 0000 1101		None	4
TBLWT*- ---	Table Write (post-decrement TBLPTR)	2	0000 0000 0000 1110		None	4
TBLWT*+ ---	Table Write (pre-increment TBLPTR)	2	0000 0000 0000 1111		None	4
XORLW k	Exclusive OR literal with W	1	0000 1010 kkkk kkkk		N,Z	

Legend: Refer to Table 3-6 for opcode field descriptions.

Note 1: 2's Complement method.

2: Unsigned arithmetic.

3: If a = '0', the Bank Select Register (BSR) will be overridden and Virtual bank is selected: If a = '1', the BSR is used.

4: Multiple cycle instruction for EPROM programming when table pointer selects internal EPROM. The instruction is terminated by an interrupt event. Writing to external program memory is a two-cycle instruction.

5: Two-cycle instruction when condition is true, else single cycle instruction.

6: Two-cycle instruction except for MOVFF to PCL (program counter low byte) in which case it takes 3 cycles.

7: A "skip" means that the instruction fetched during execution of the current instruction is not executed, instead an NOP is executed.

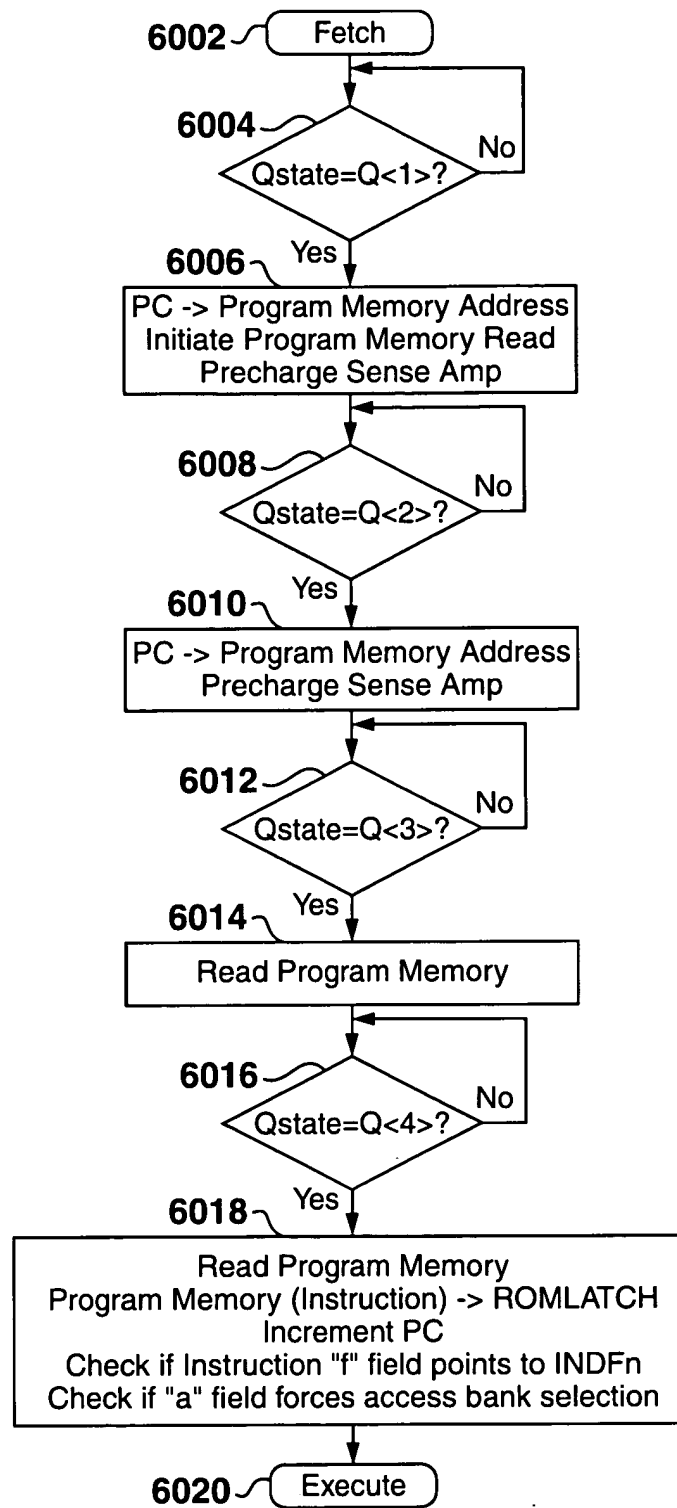
8: When a "skip" instruction executes a skip and is followed by a 2-word instruction, 3 cycles will be executed.

9: If s = '1', certain registers will be loaded from/into shadow registers. If s = '0' no update occurs.

Instruction Set Summary (Continued)

Figure 59

41/95



Valid For:
 ADDWF
 ADDWFC
 ANDWF
 COMF
 DECF
 INCF
 IORWF
 MOVF
 RLCF
 RLNCF
 RRCF
 RRNCF
 SUBFWB
 SUBWF
 SUBWFB
 SWAPF
 XORWF
 MOVWF
 NOP

Figure 60

42/95

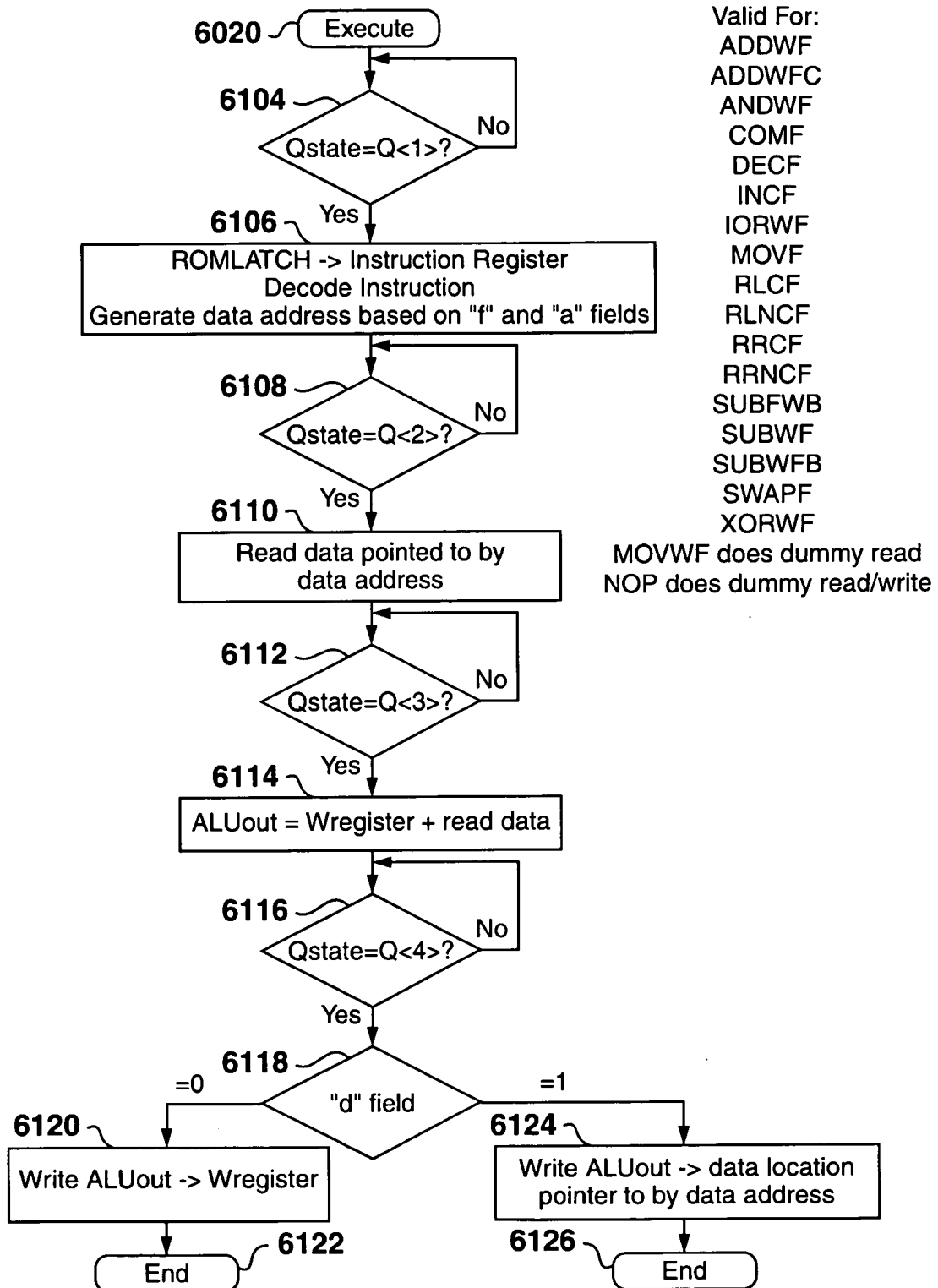


Figure 61

43/95

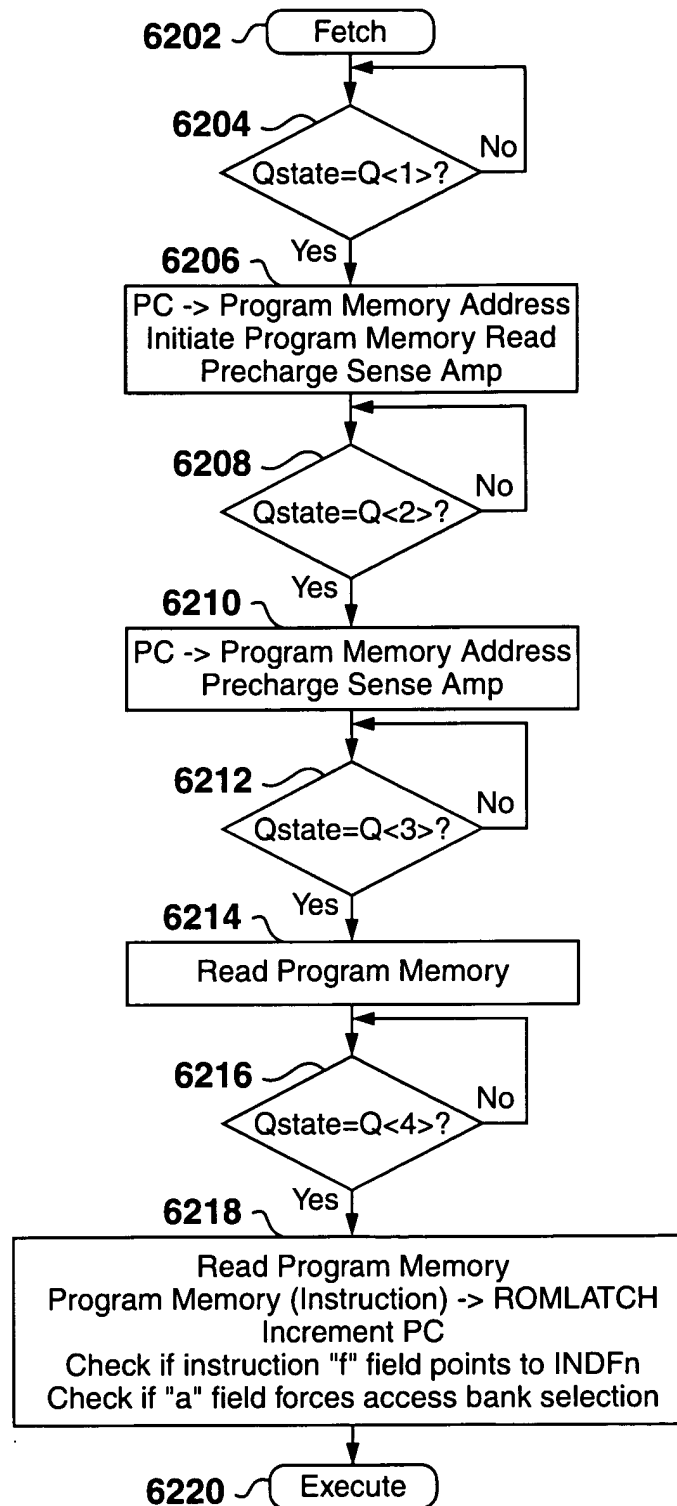


Figure 62

44/95

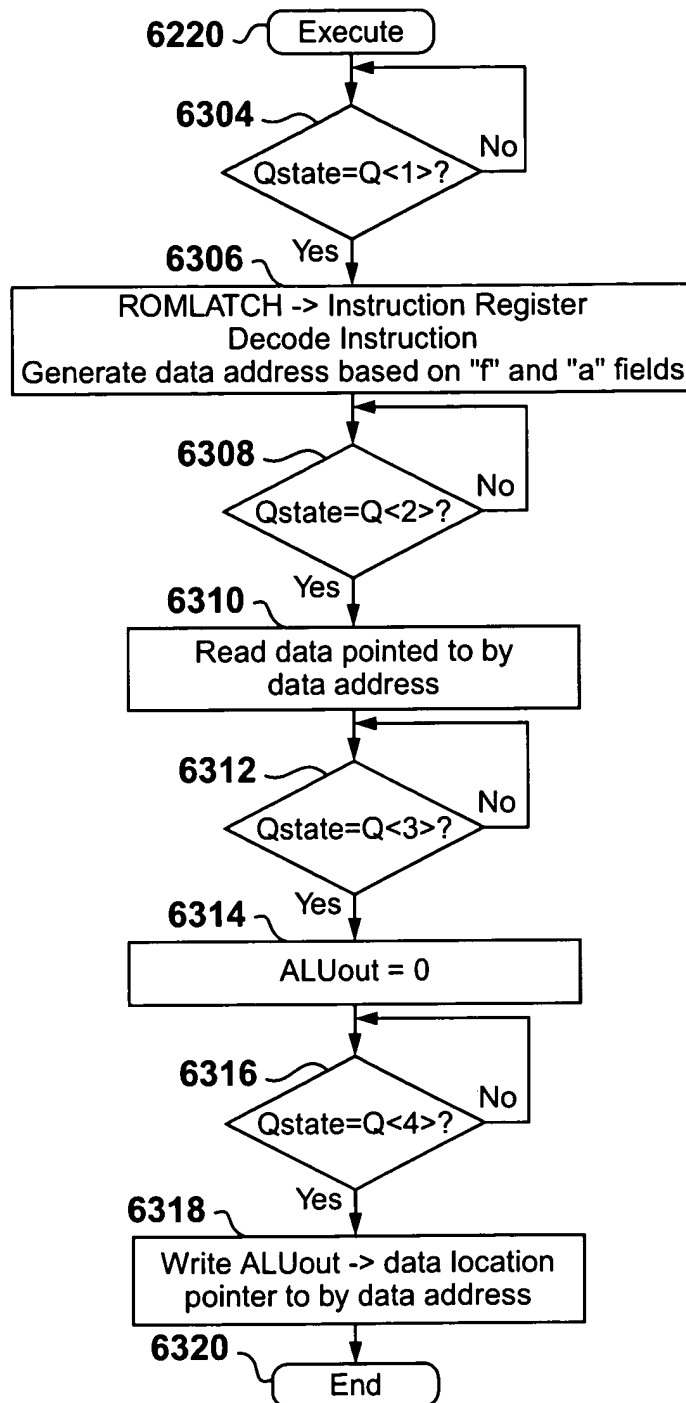


Figure 63

45/95

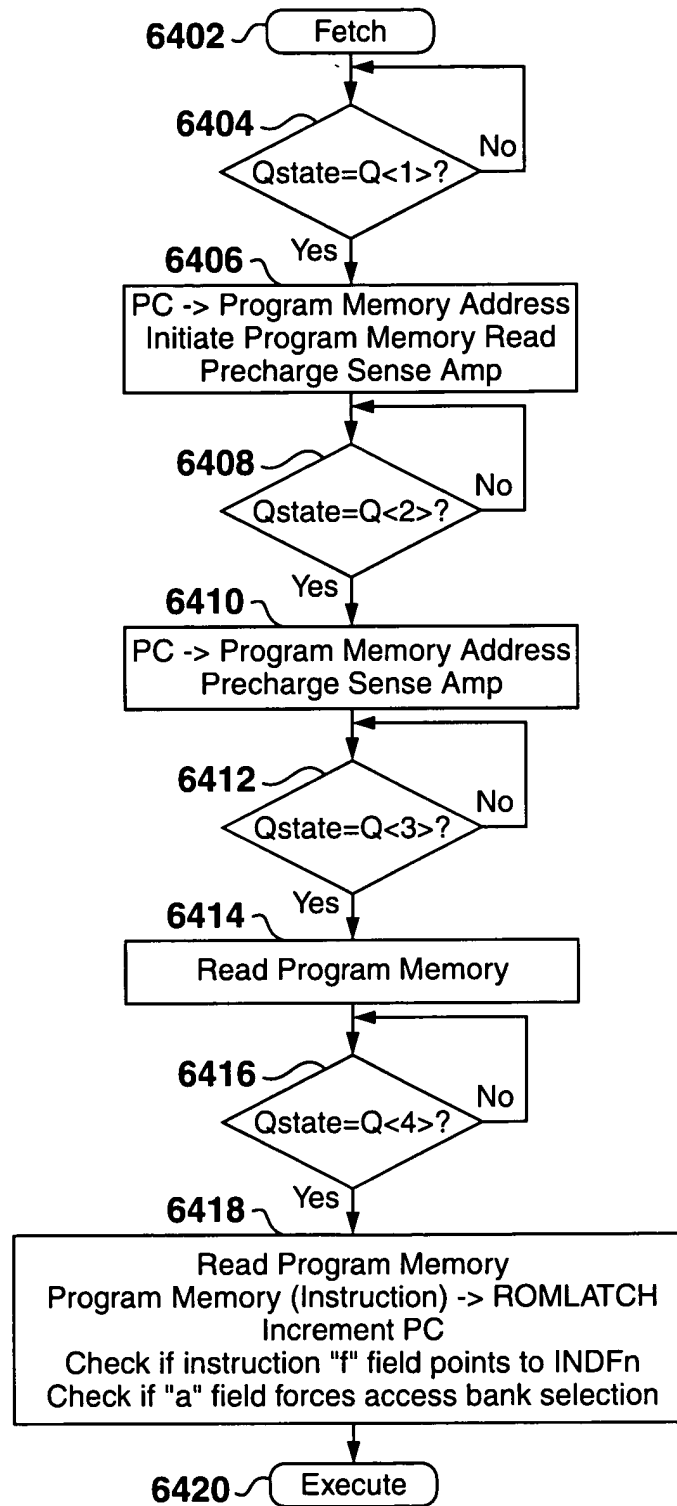


Figure 64

46/95

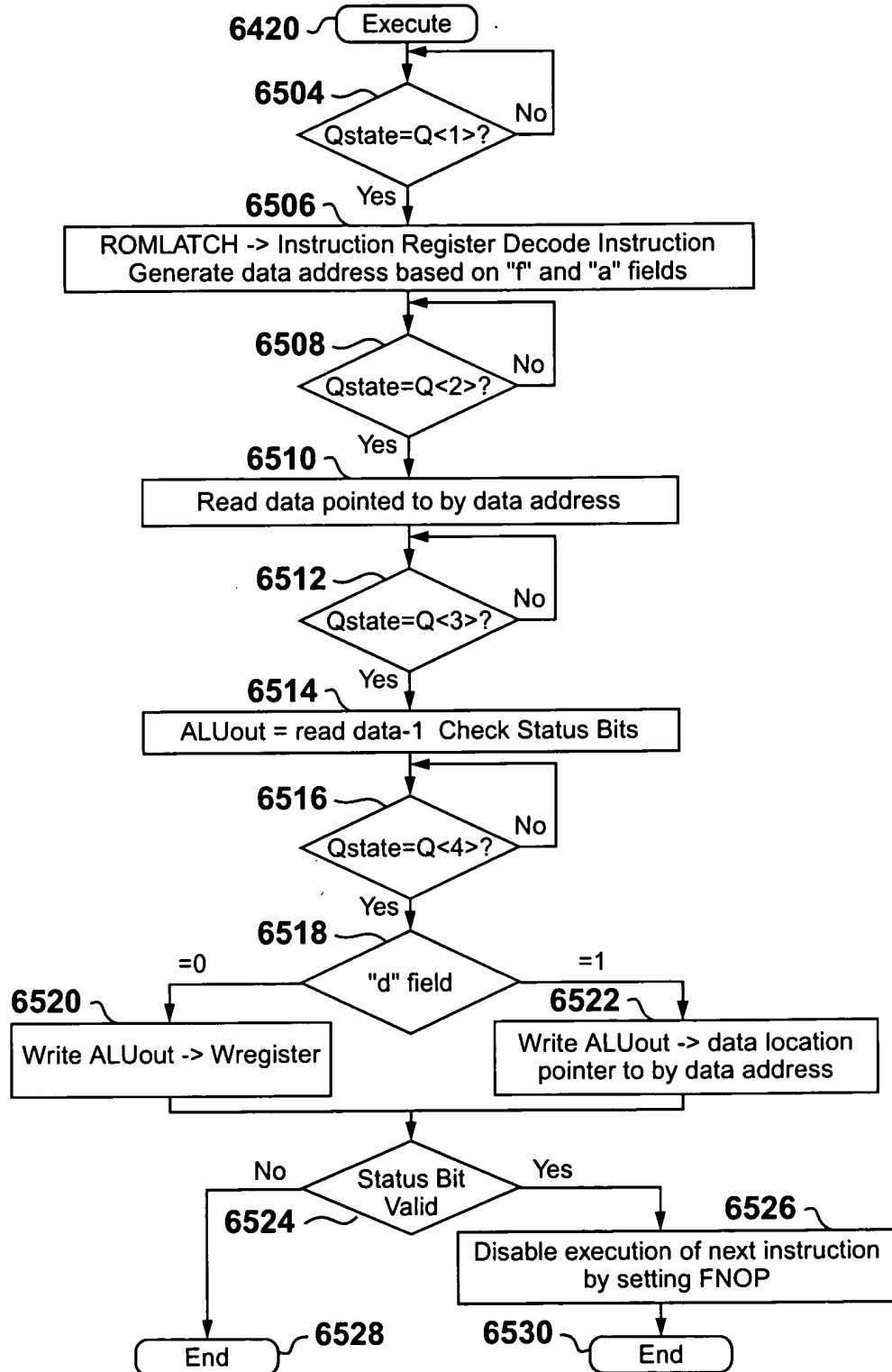


Figure 65

47/95

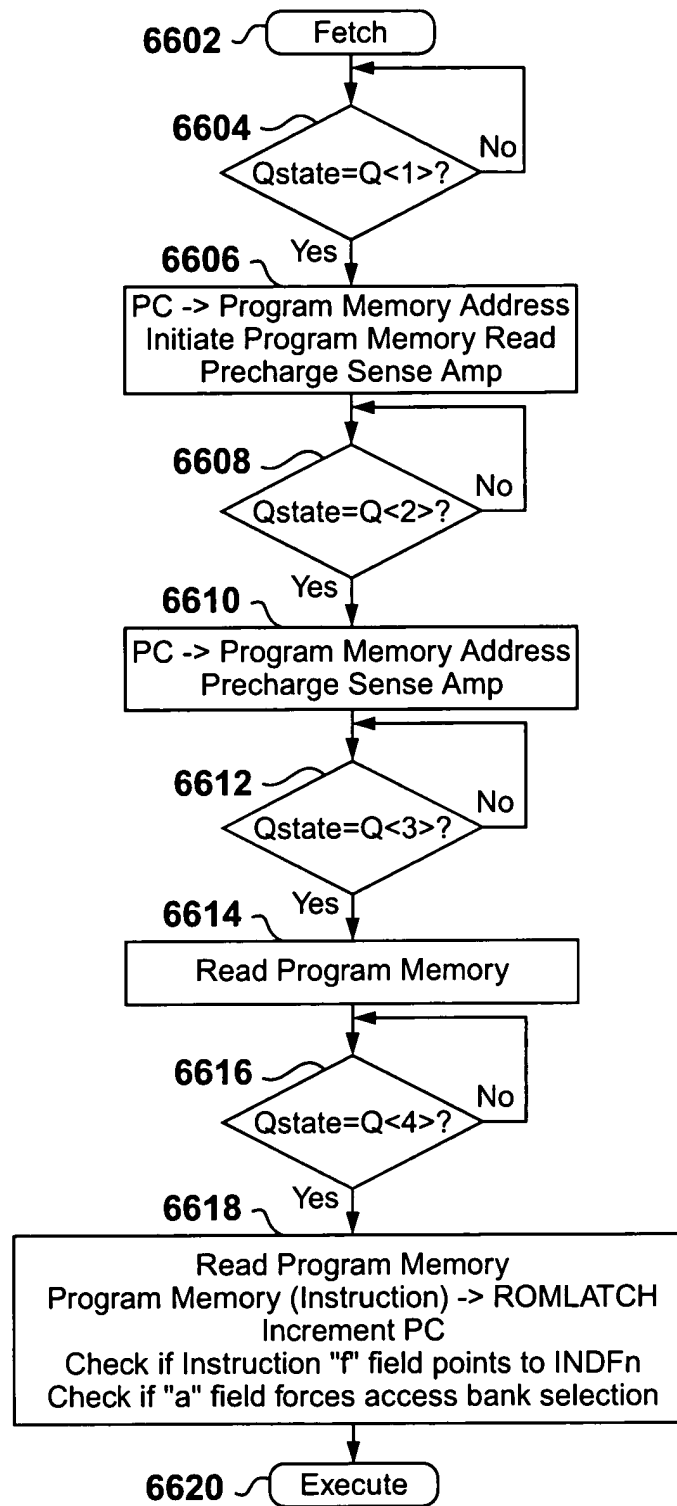


Figure 66

48/95

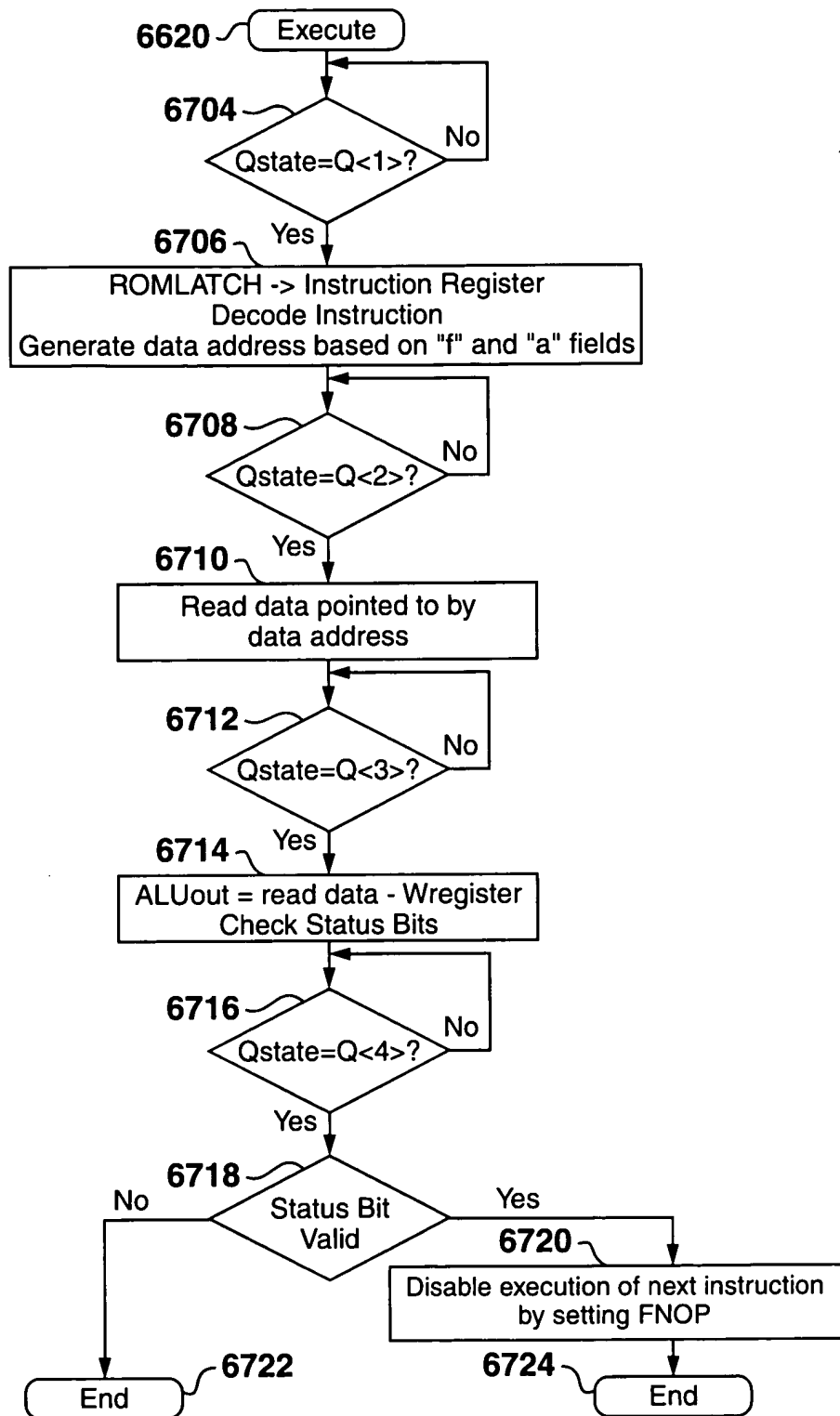


Figure 67

49/95

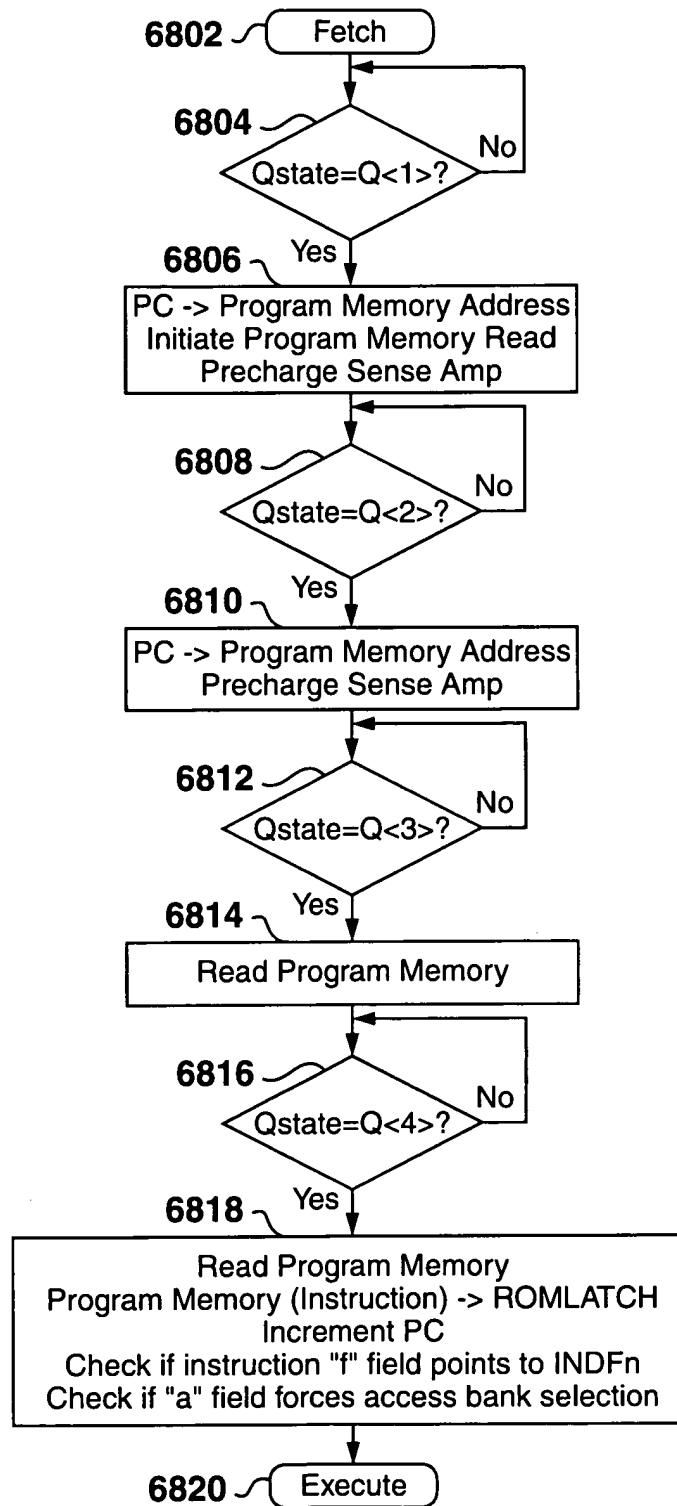


Figure 68

50/95

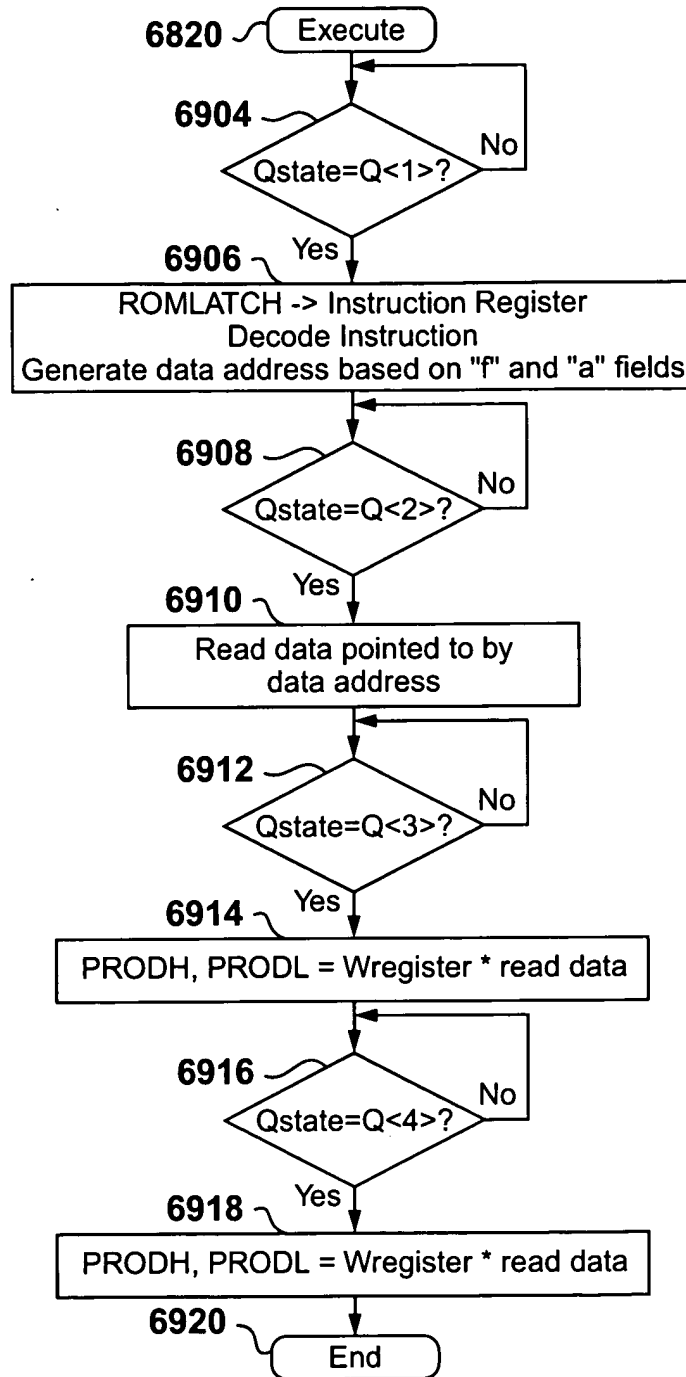


Figure 69

51/95

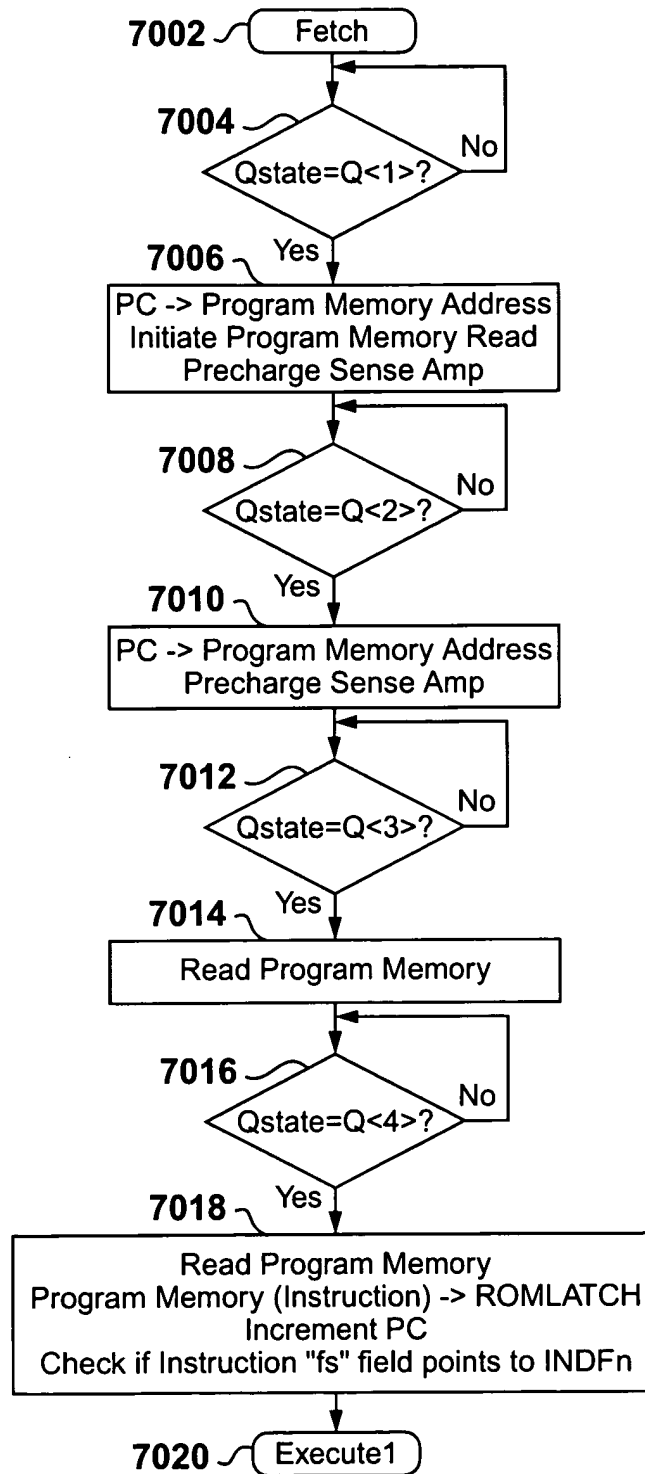


Figure 70

52/95

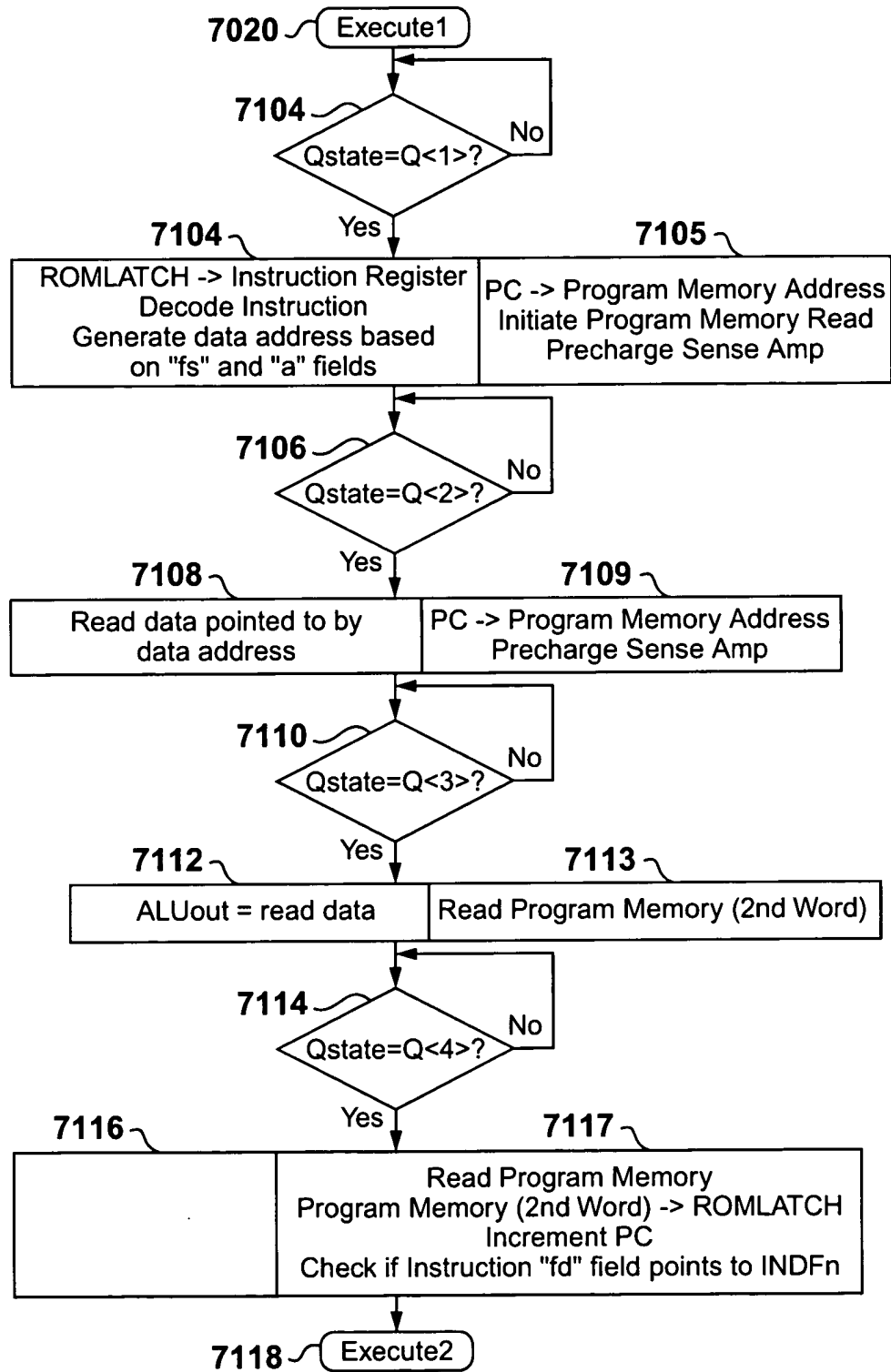


Figure 71

53/95

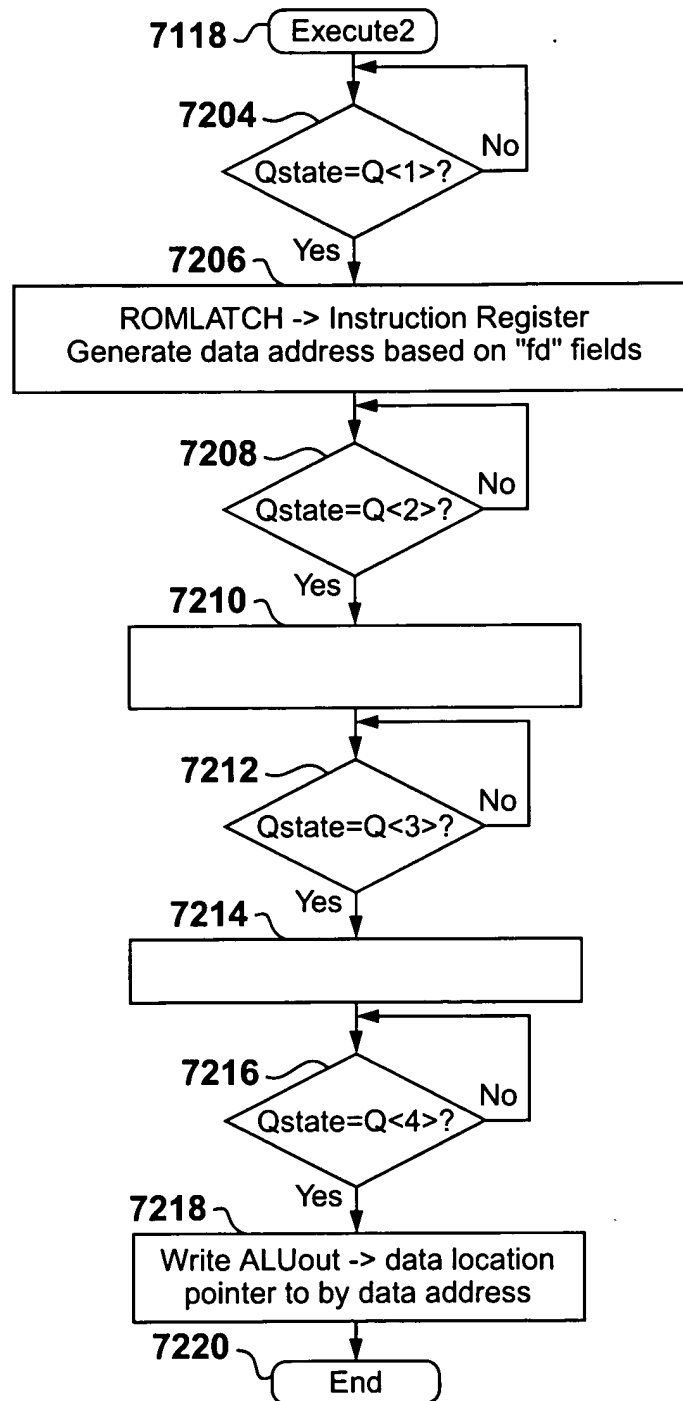


Figure 72

54/95

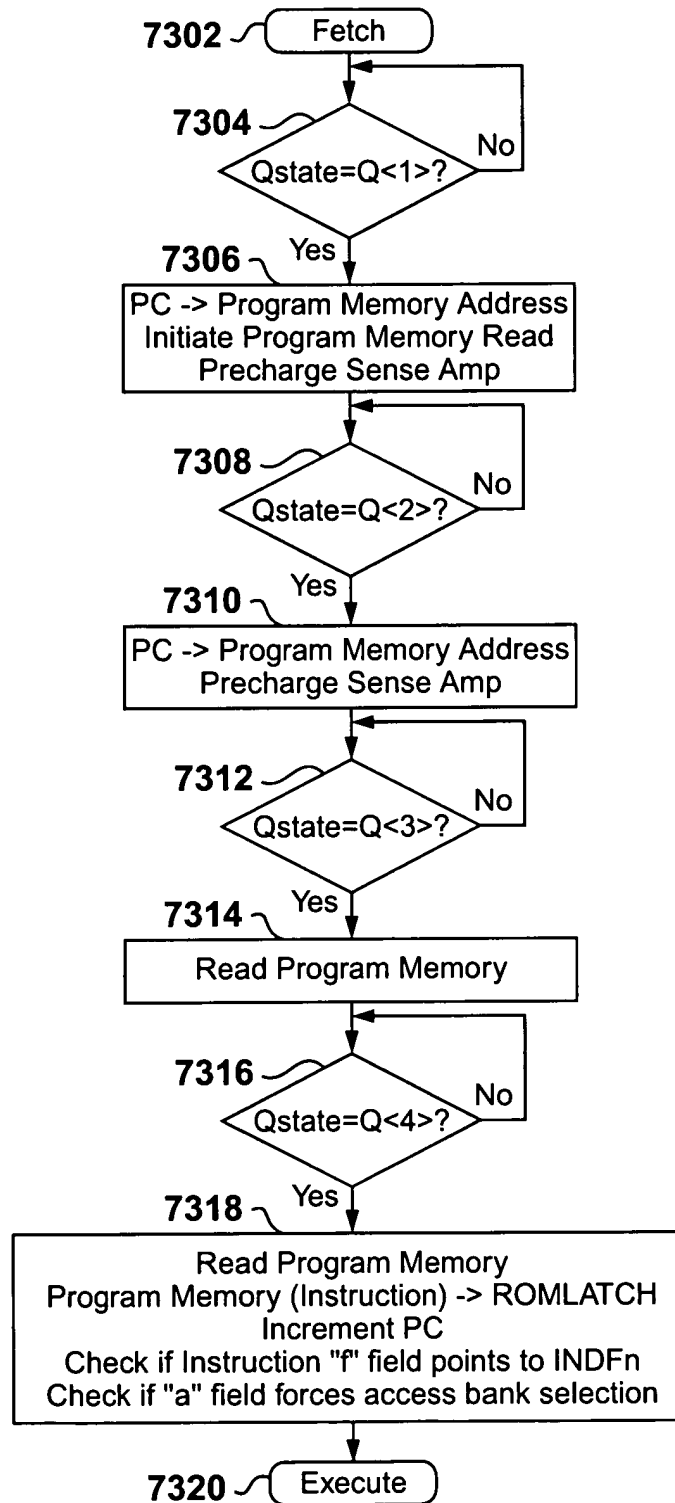


Figure 73

55/95

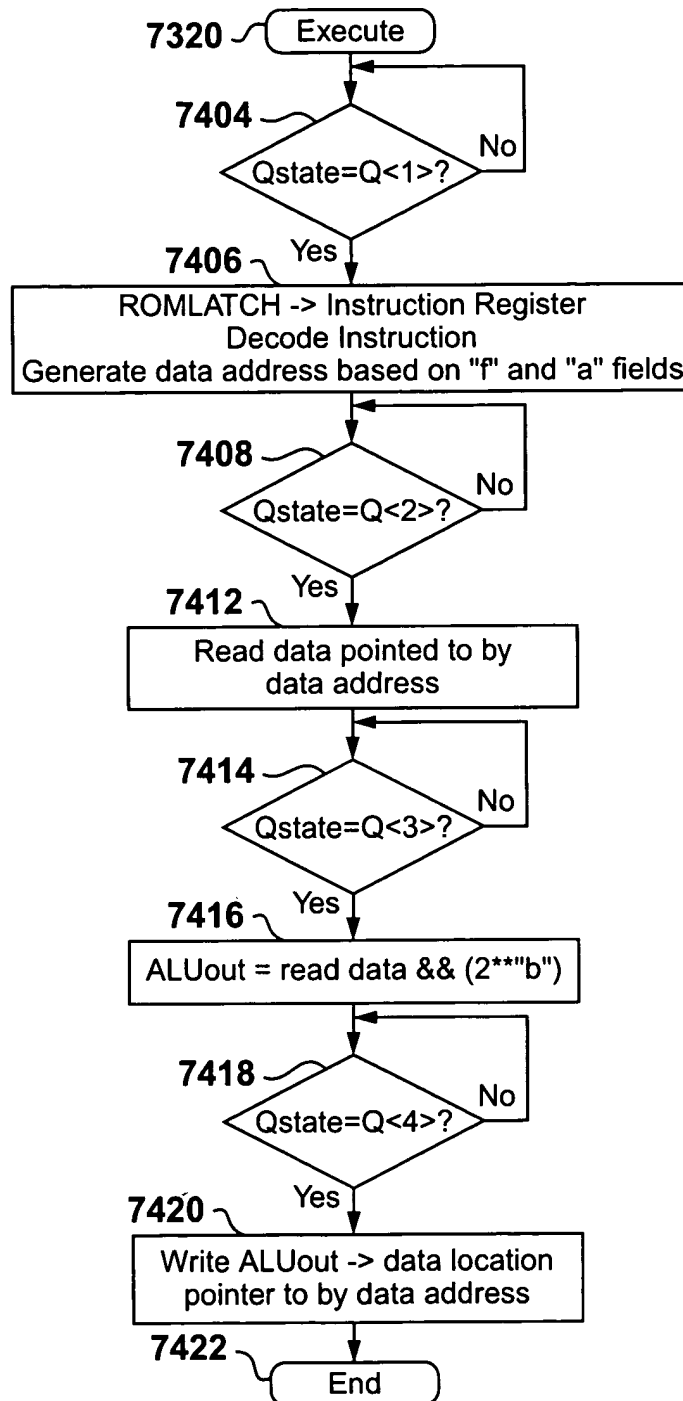


Figure 74

56/95

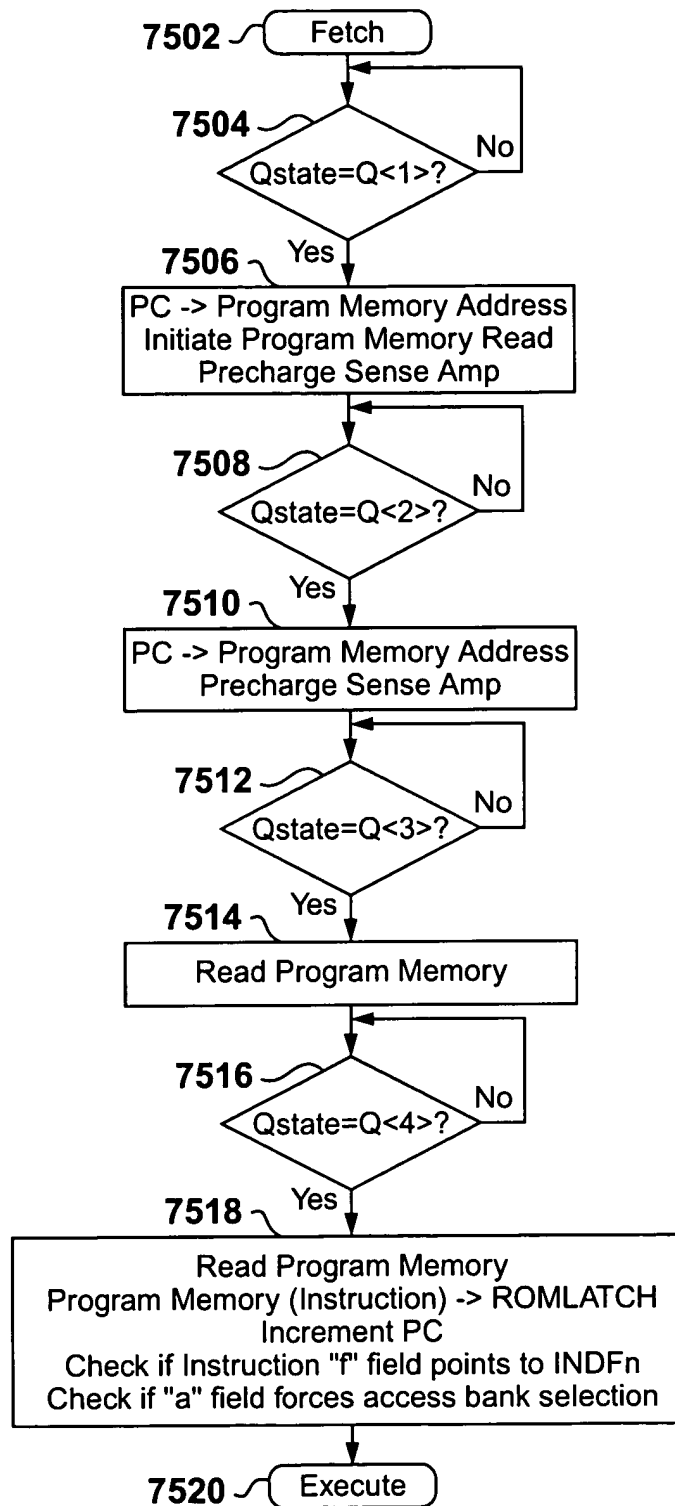


Figure 75

57/95

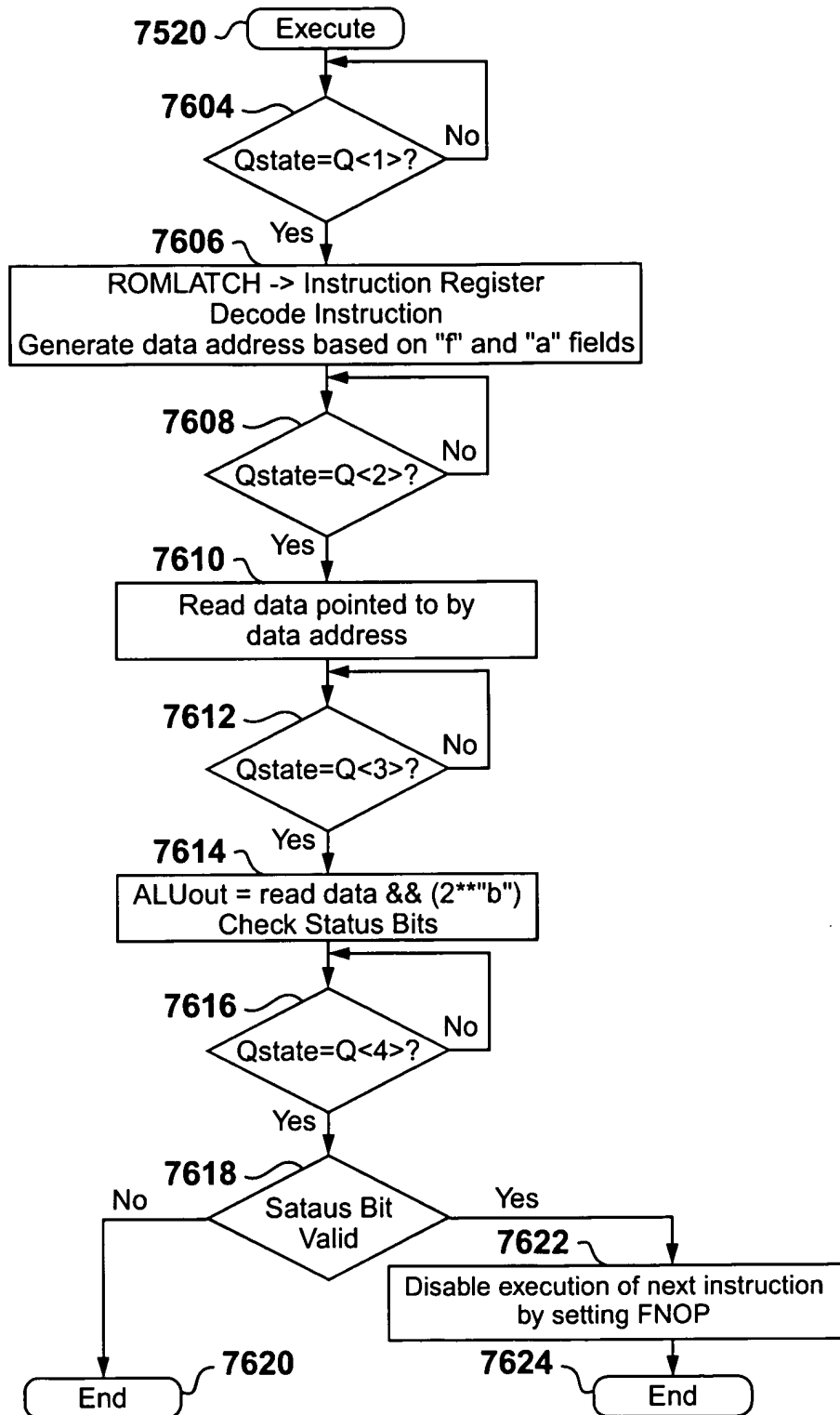


Figure 76

58/95

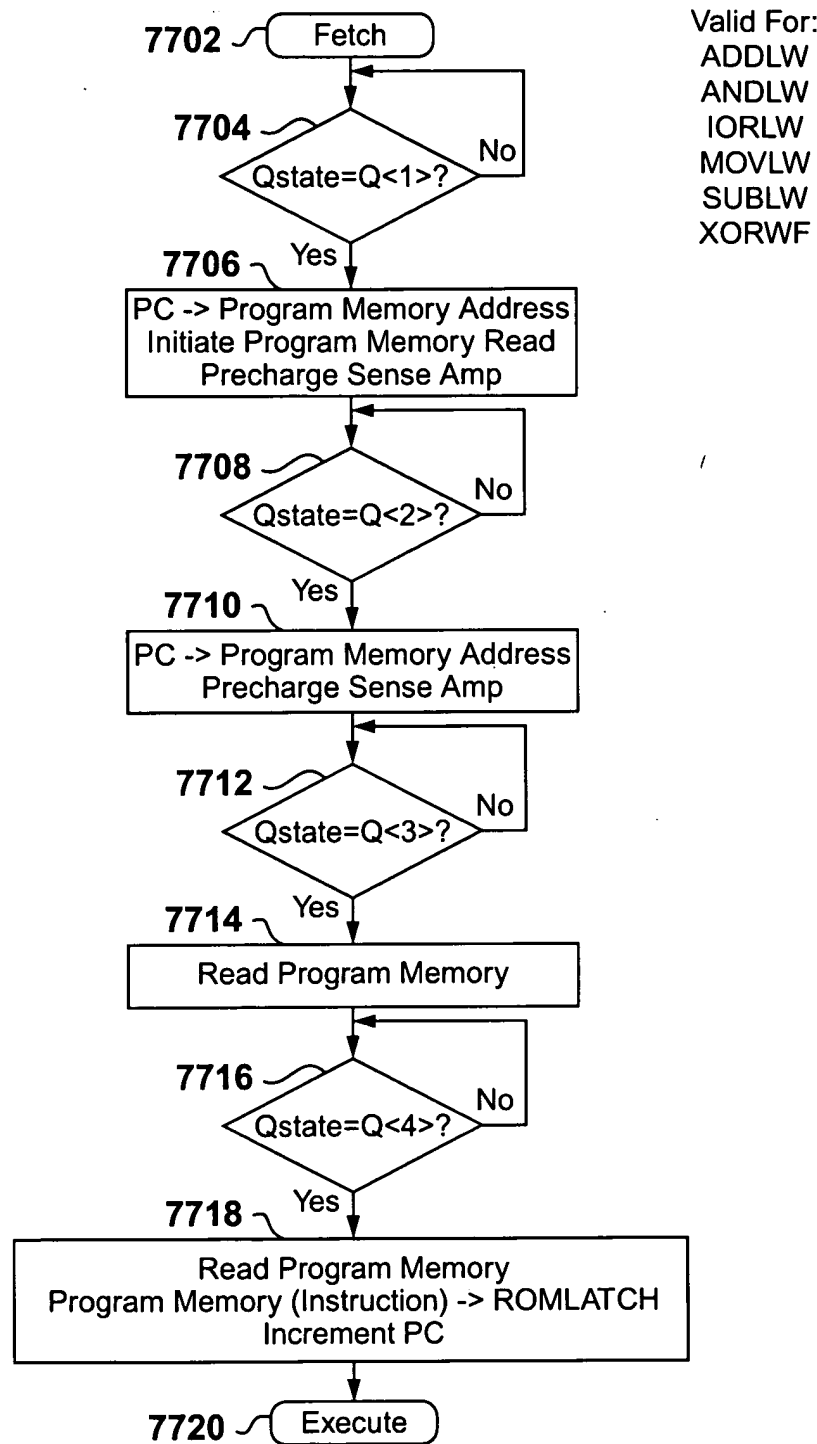
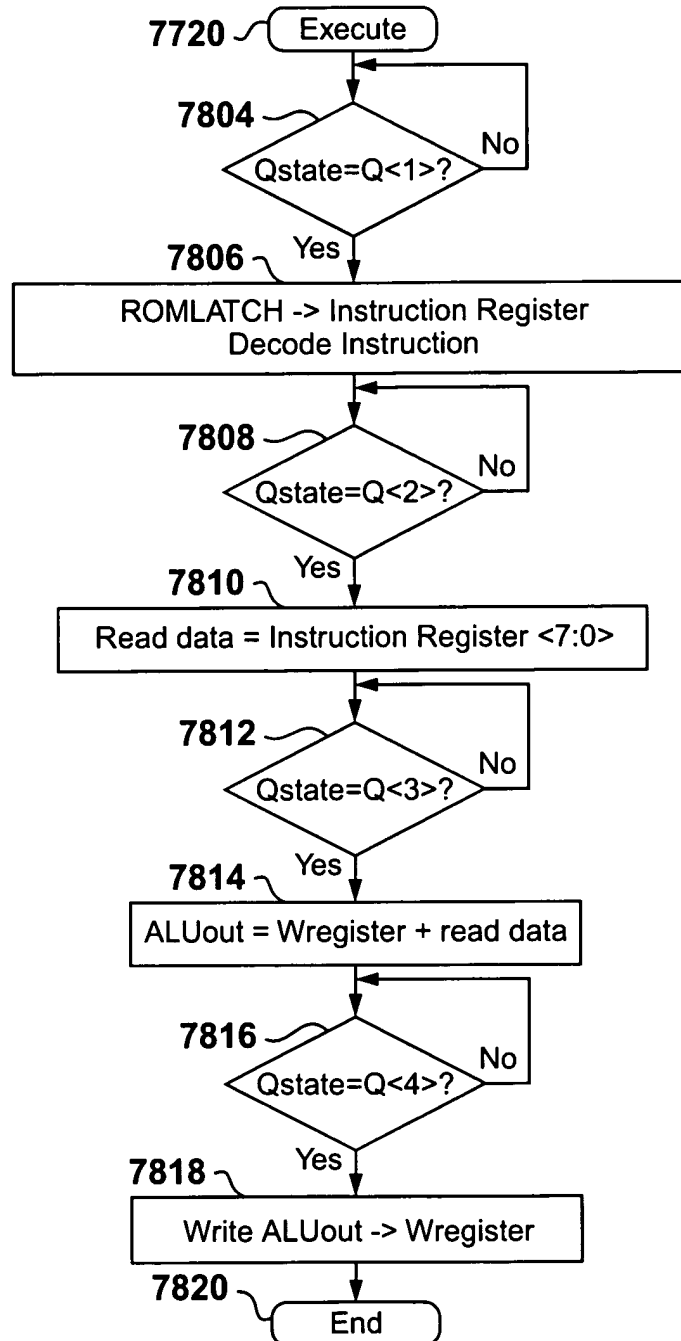


Figure 77

59/95



Valid For:
ADDLW
ANDLW
IORLW
MOVLW
SUBLW
XORLW

Figure 78

60/95

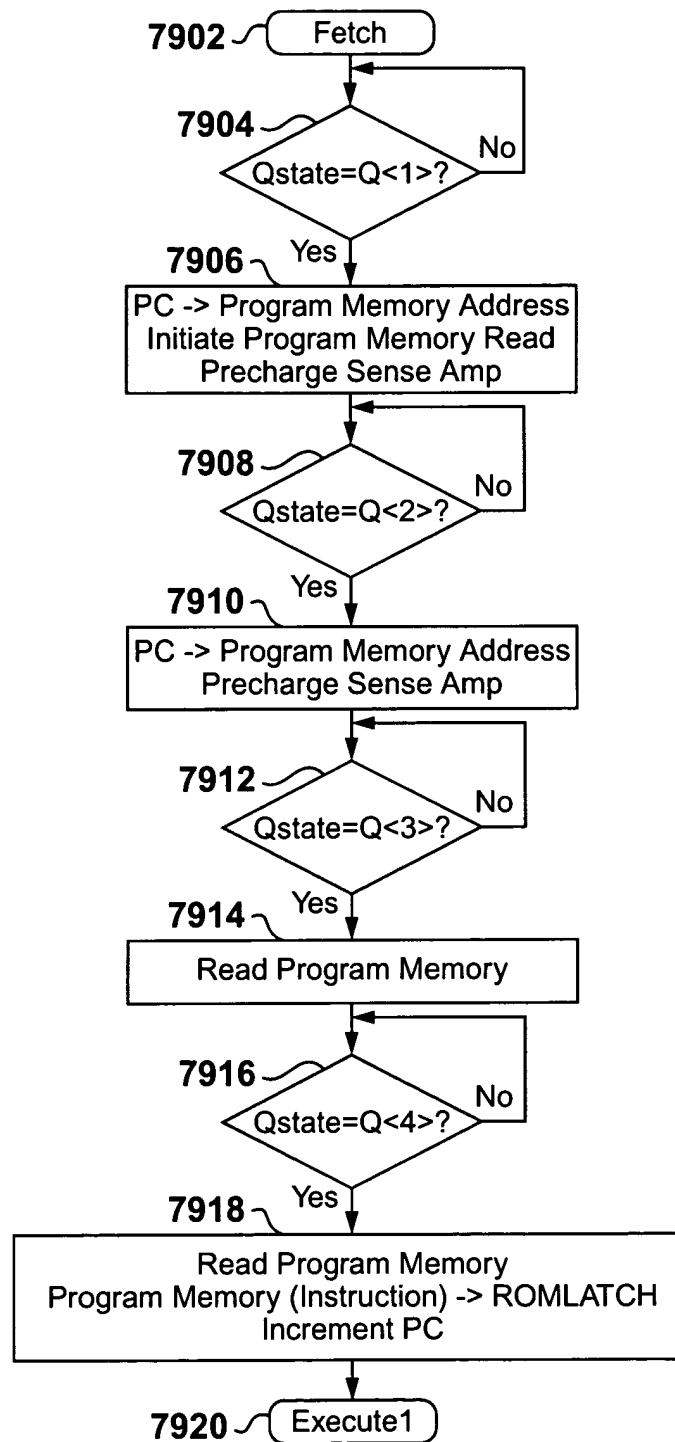


Figure 79

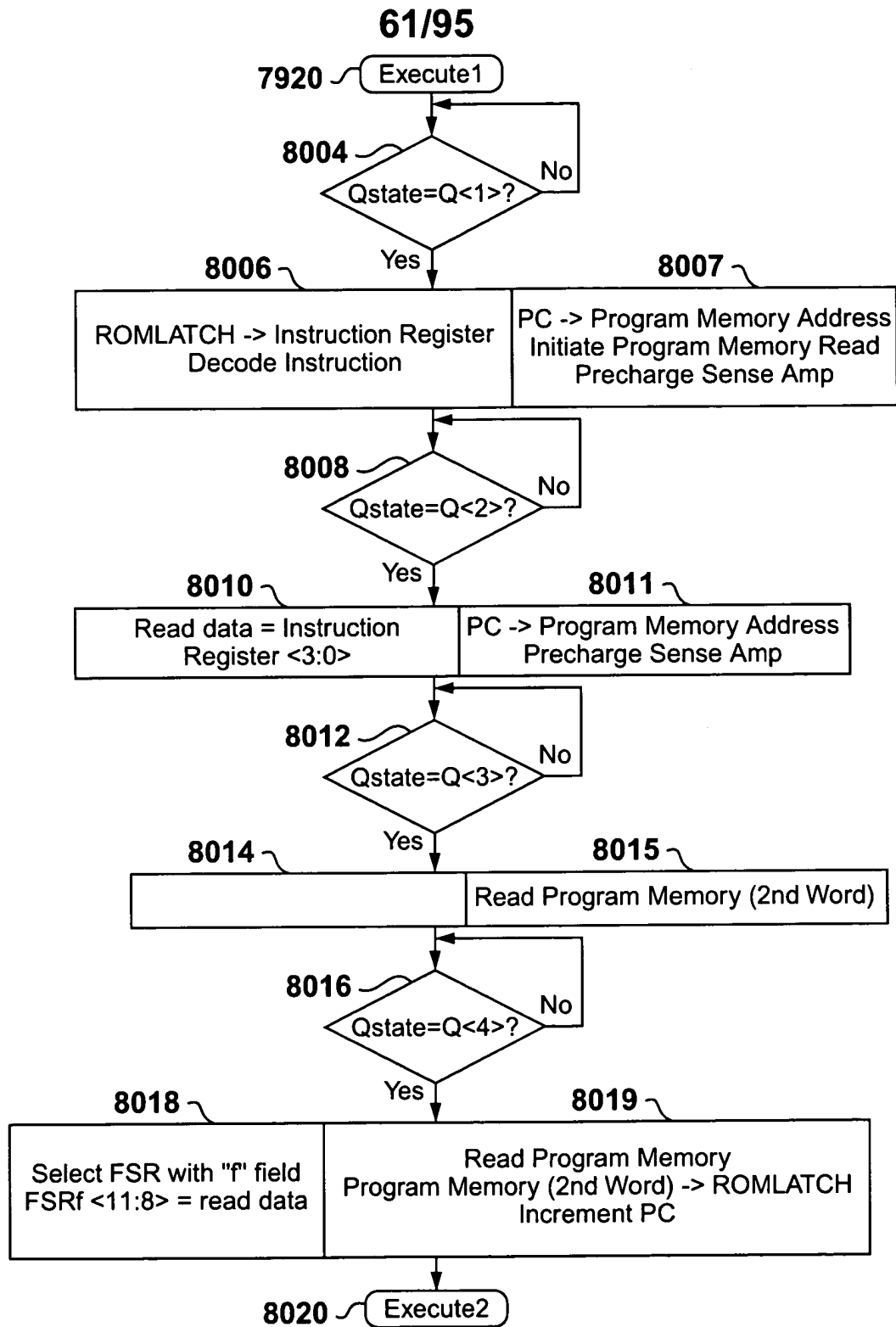


Figure 80

62/95

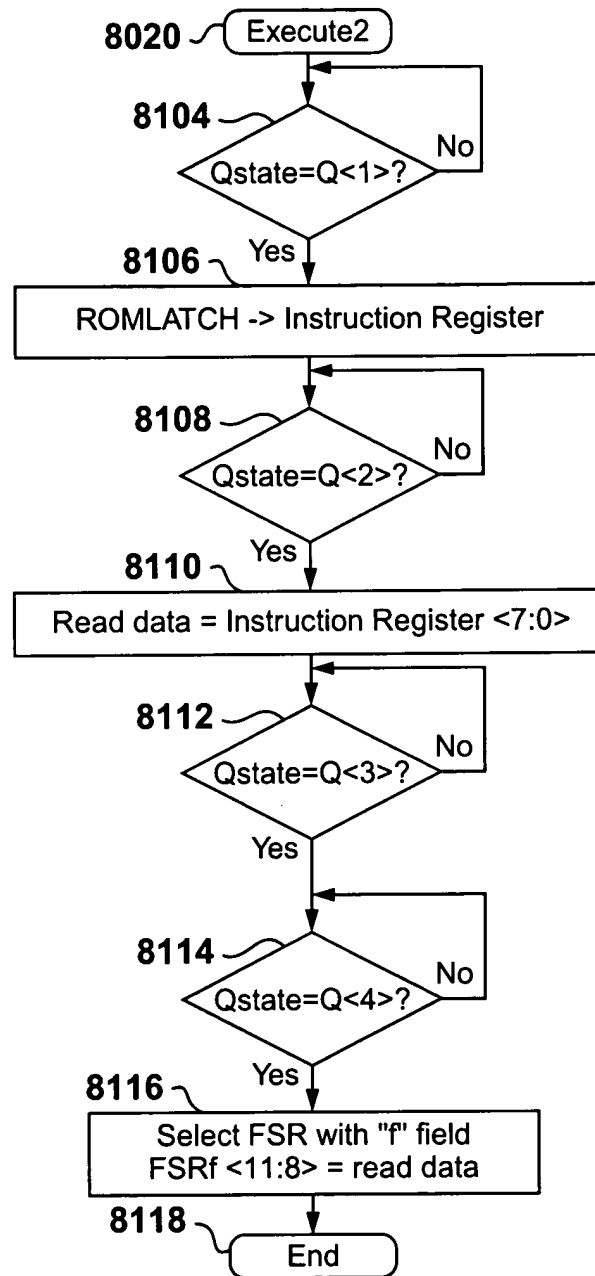


Figure 81

63/95

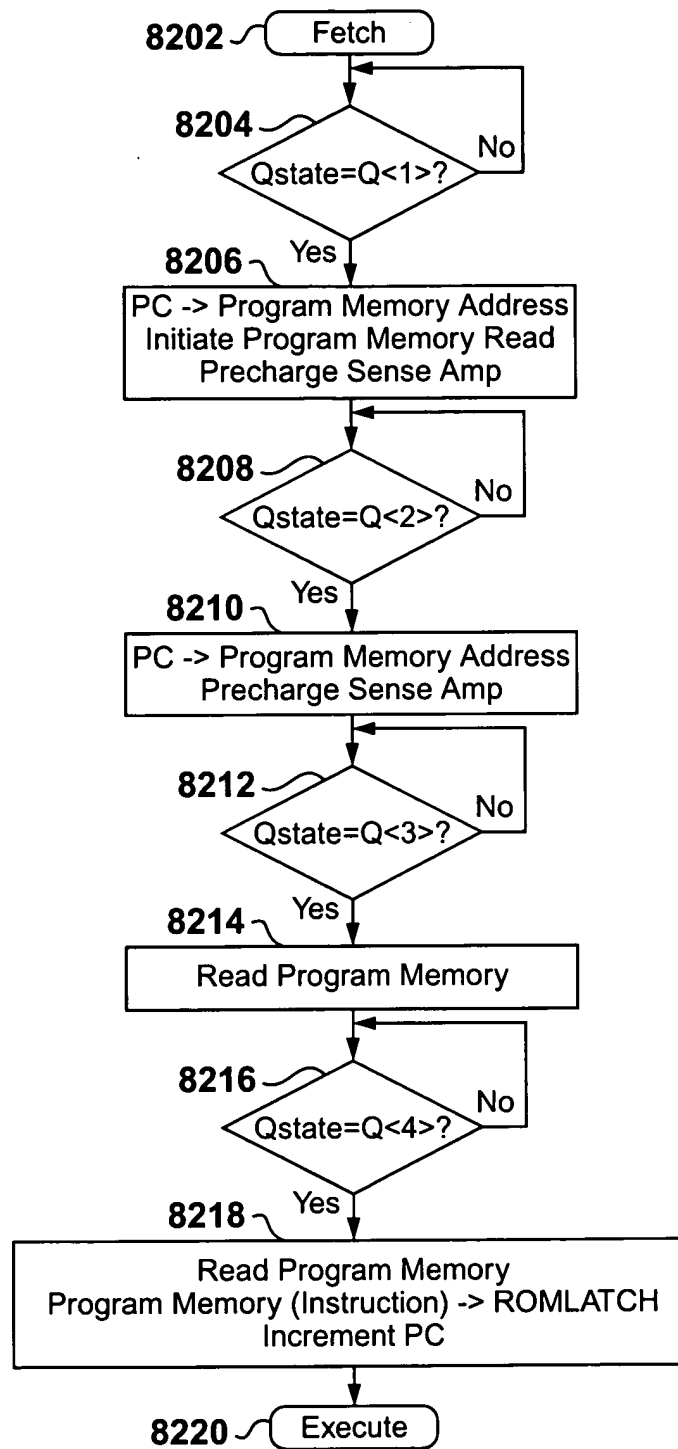


Figure 82

64/95

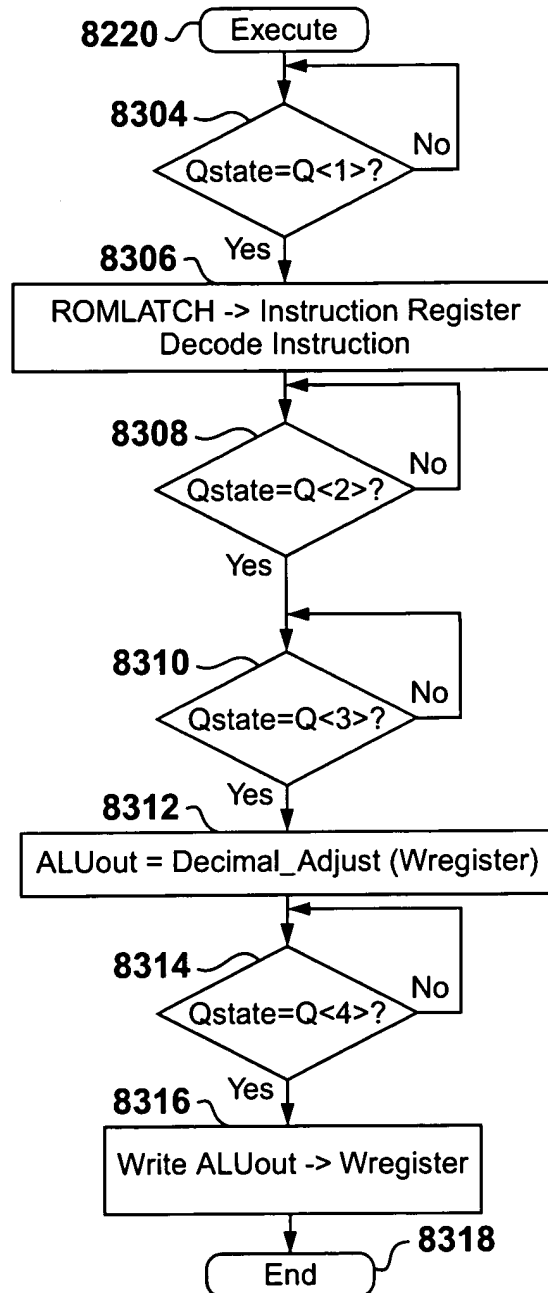


Figure 83

65/95

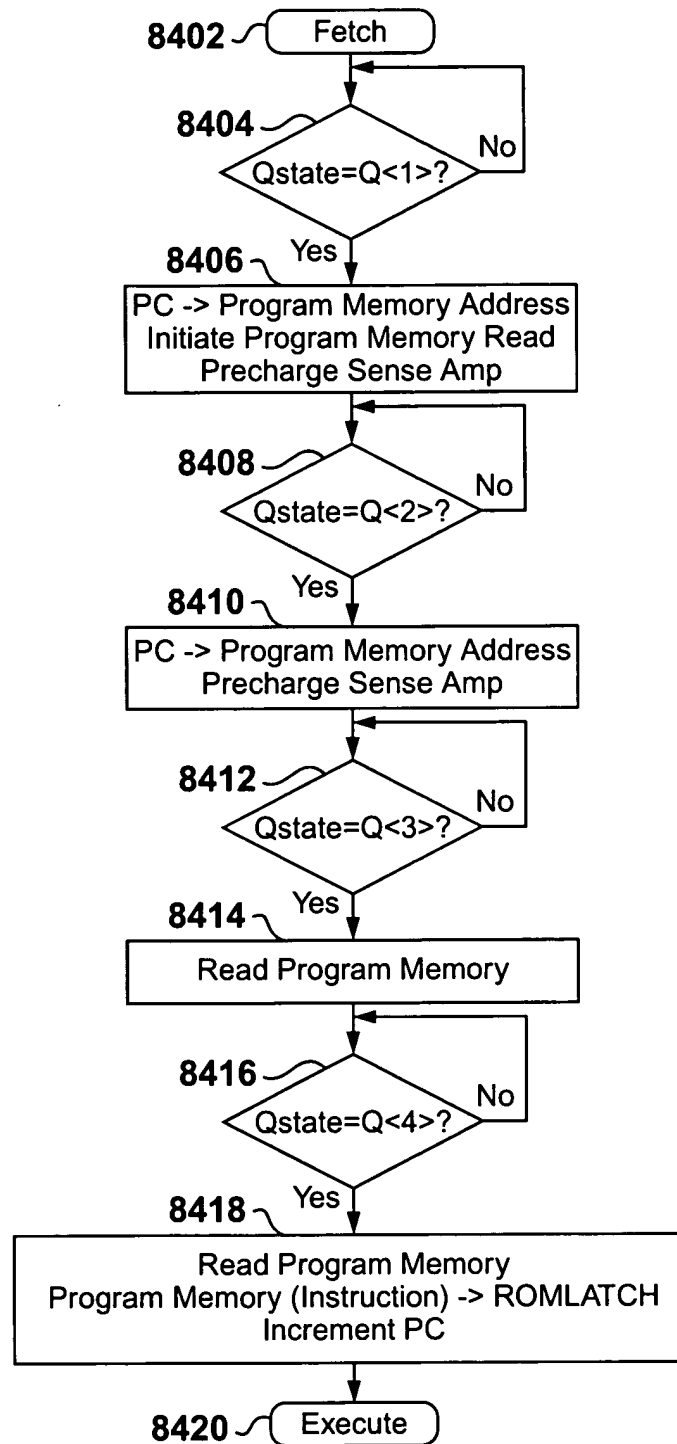


Figure 84

66/95

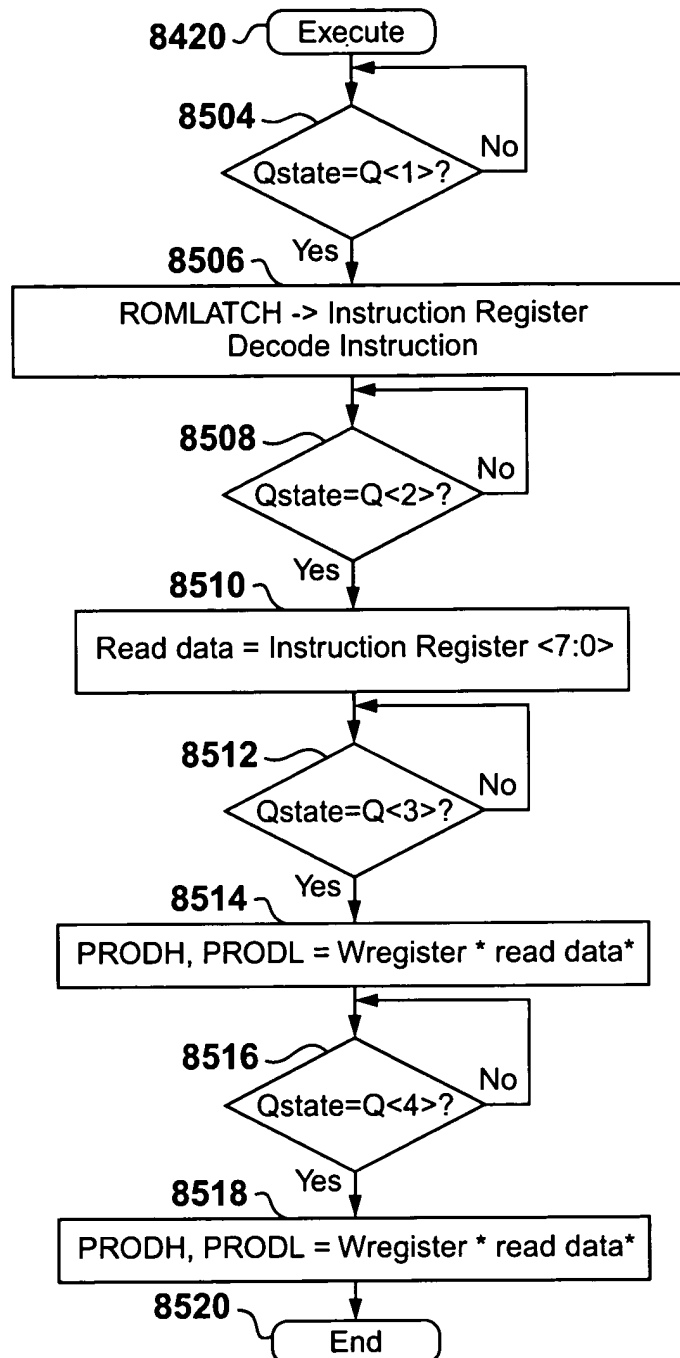


Figure 85

67/95

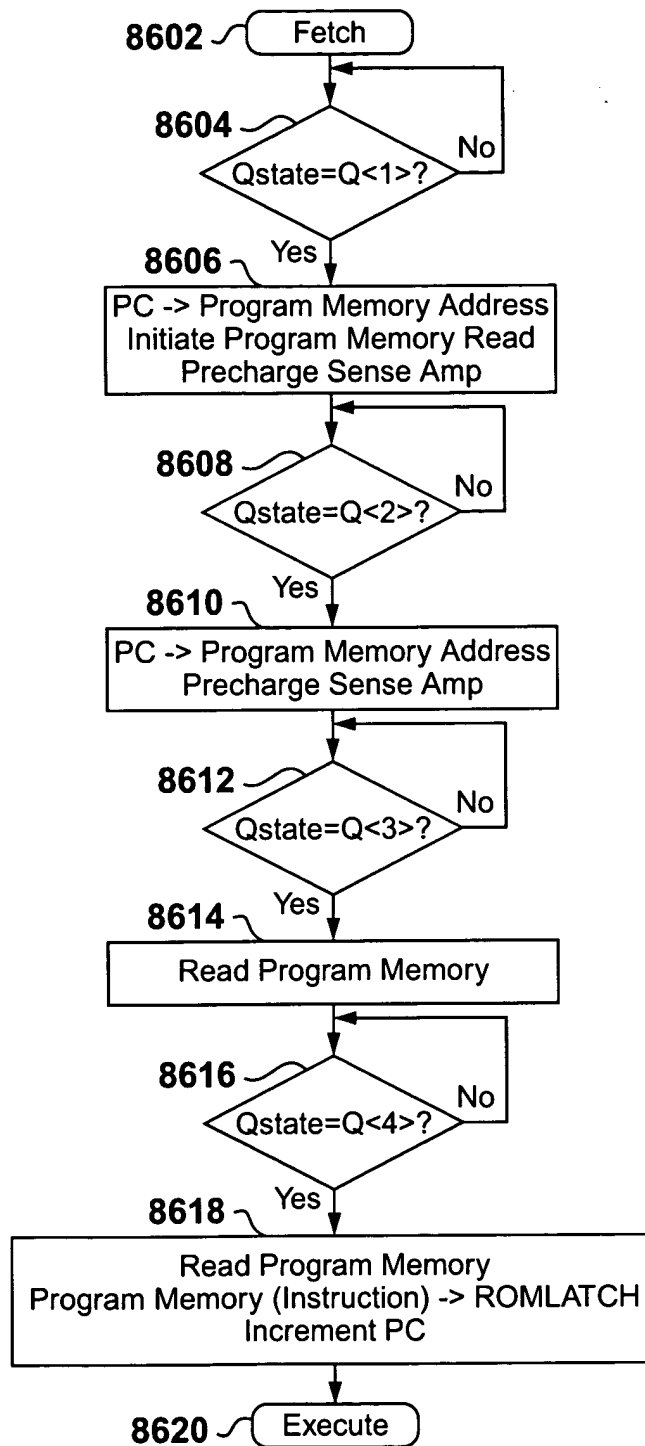


Figure 86

68/95

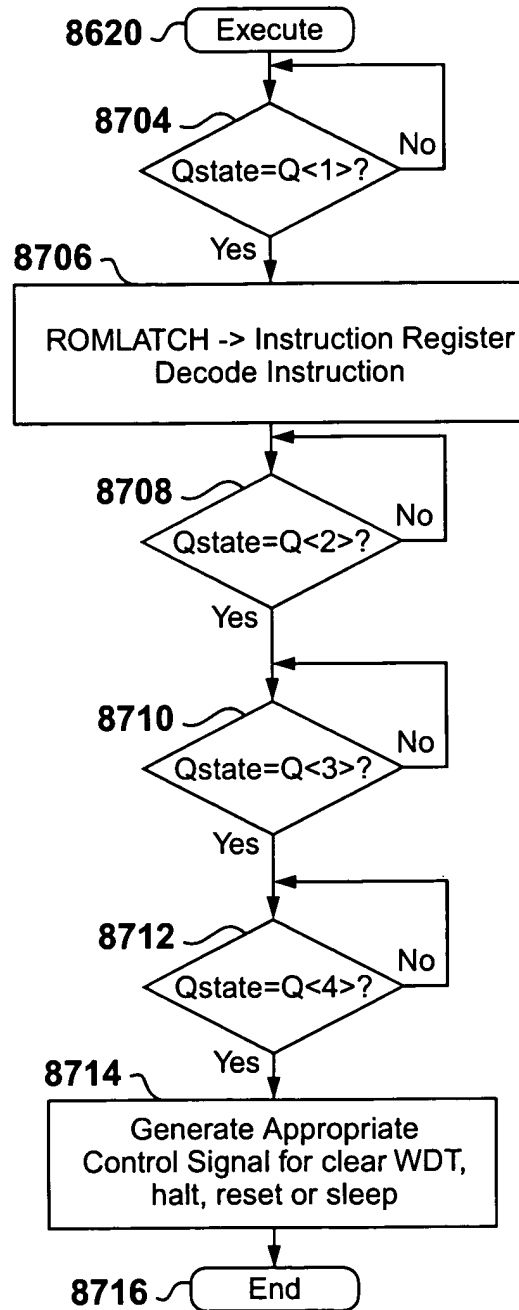


Figure 87

69/95

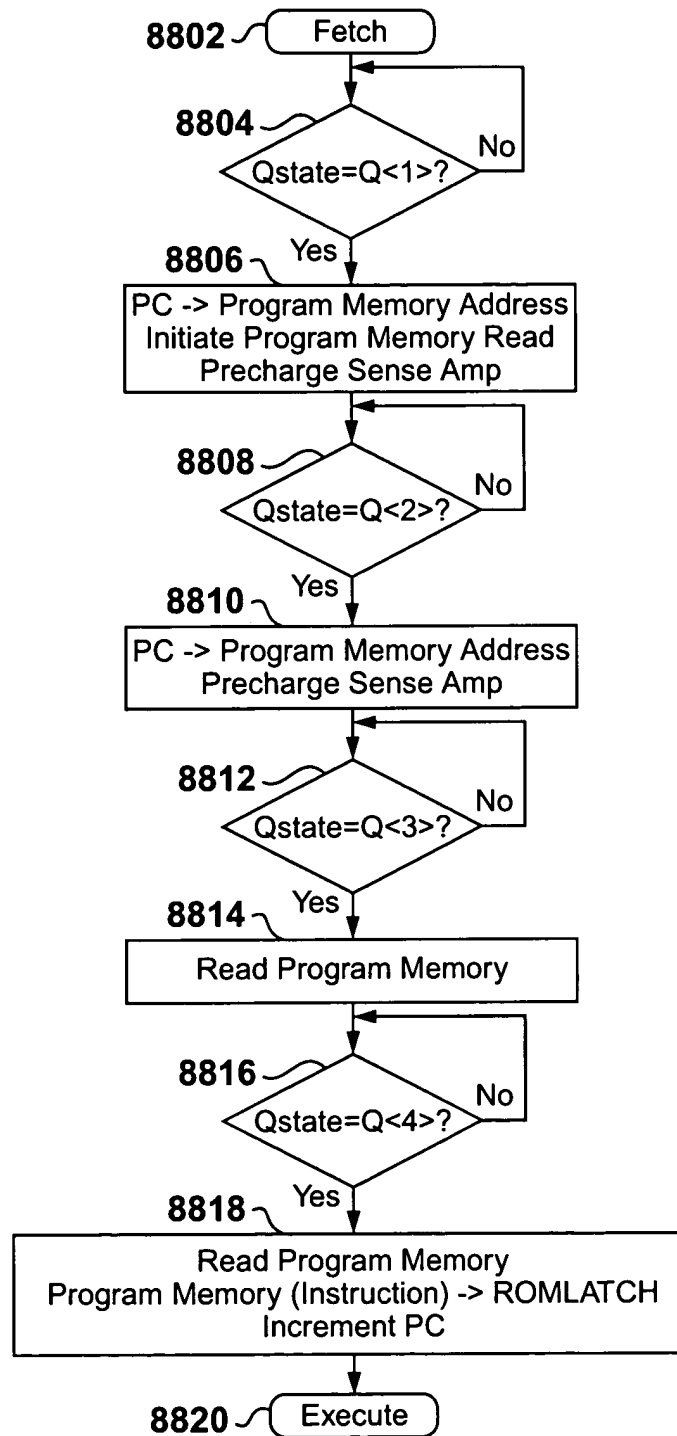


Figure 88

70/95

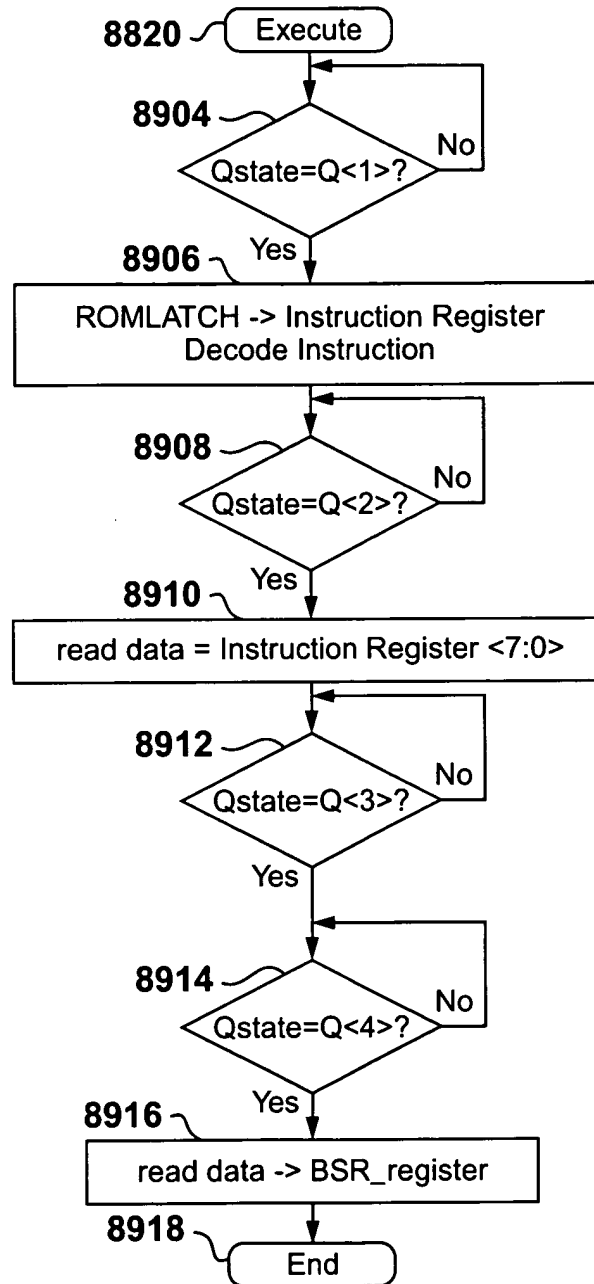


Figure 89

71/95

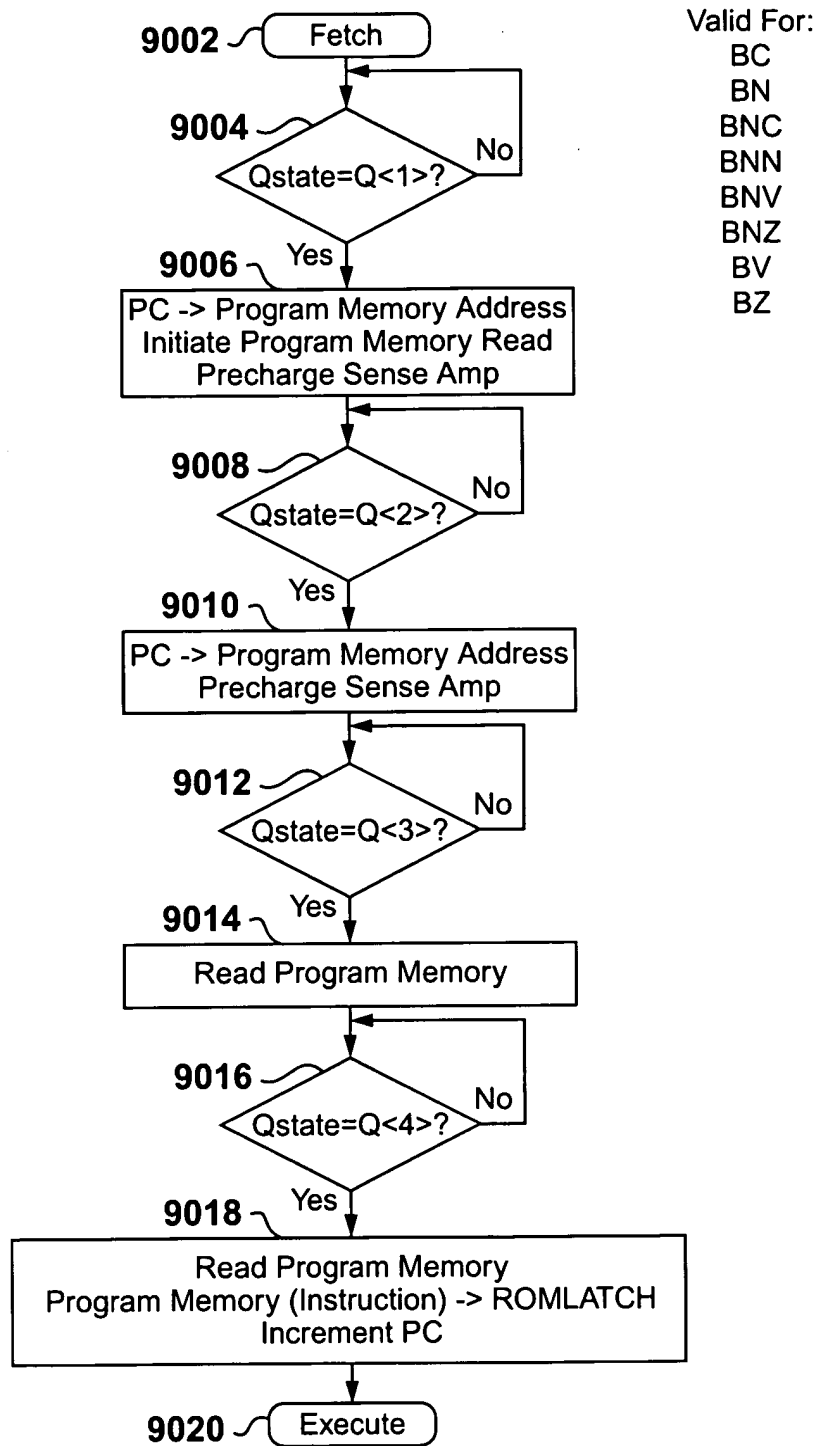


Figure 90

72/95

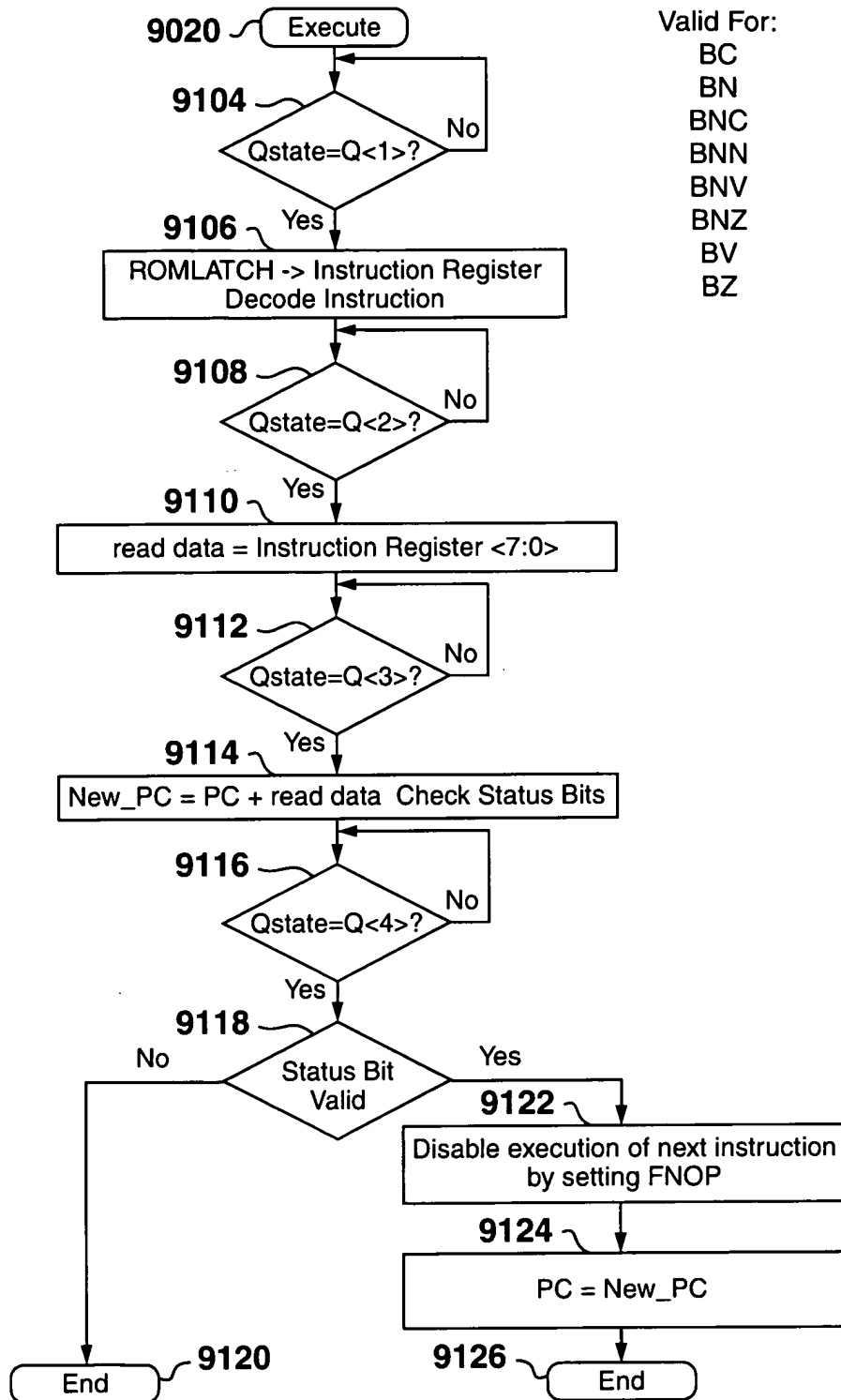


Figure 91

73/95

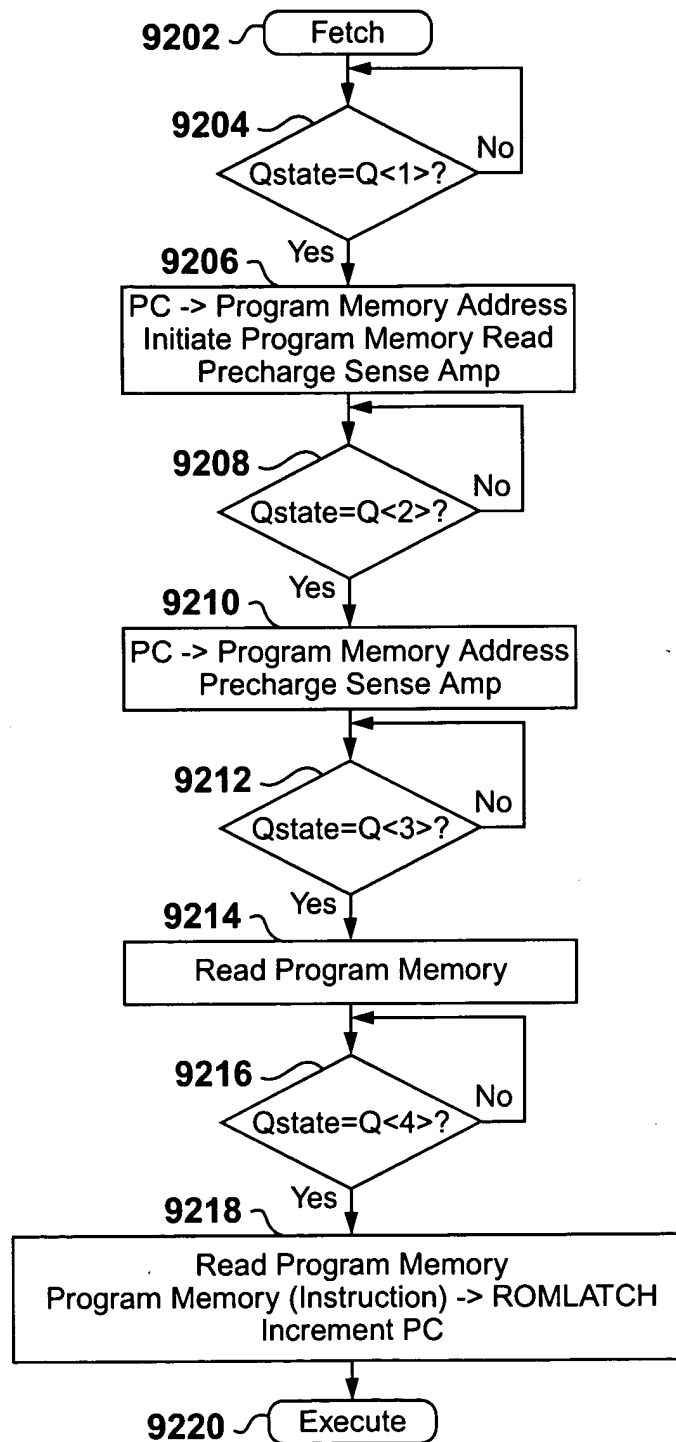


Figure 92

74/95

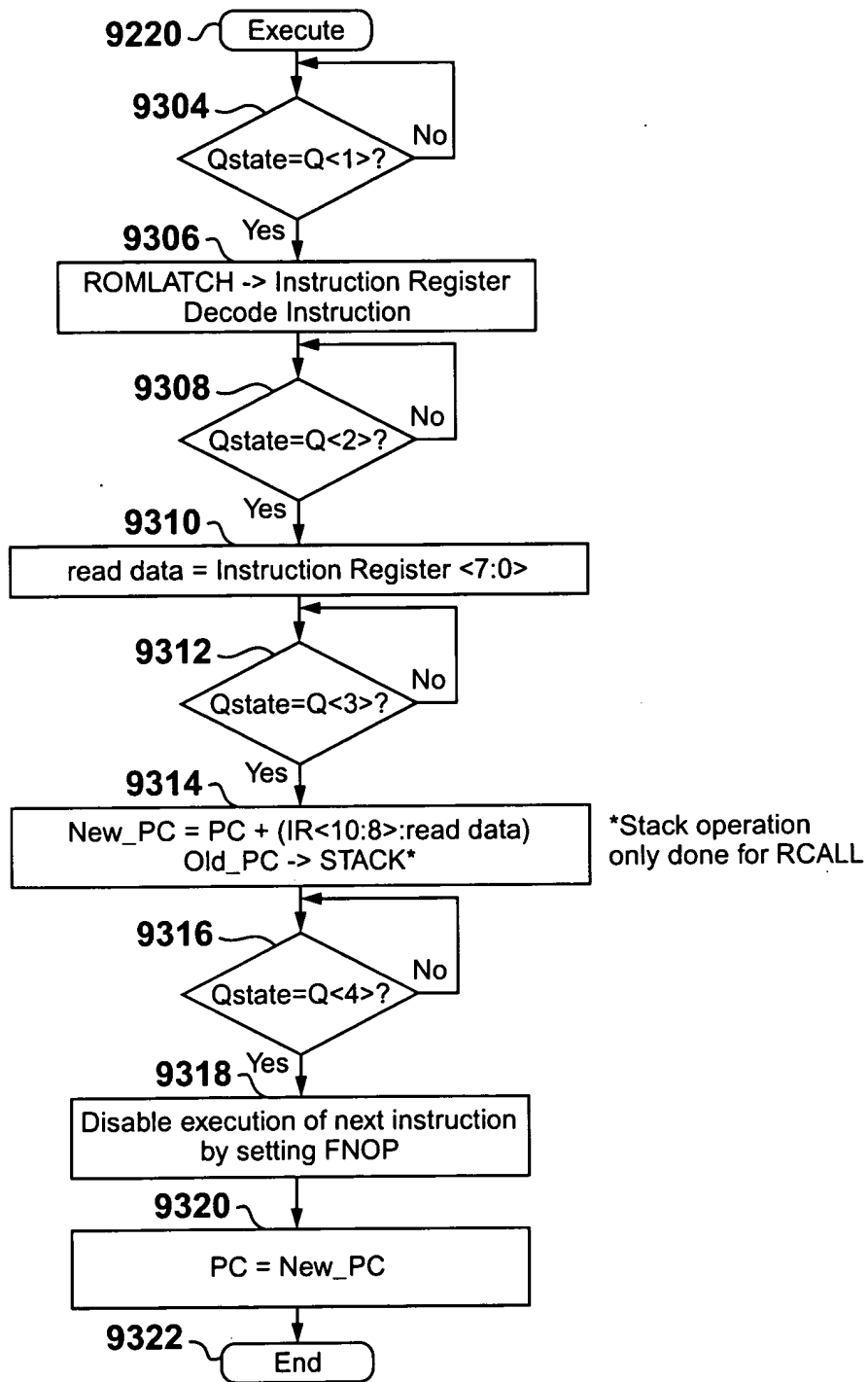


Figure 93

75/95

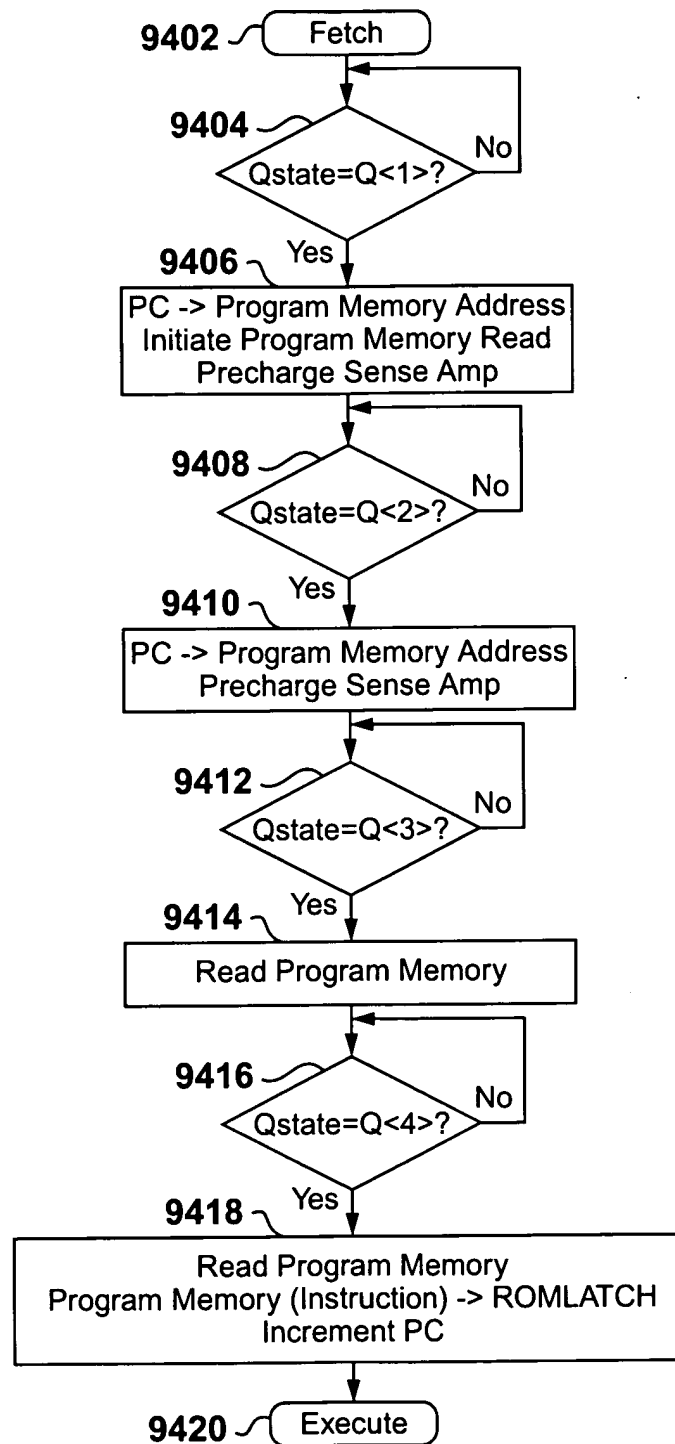


Figure 94

76/94

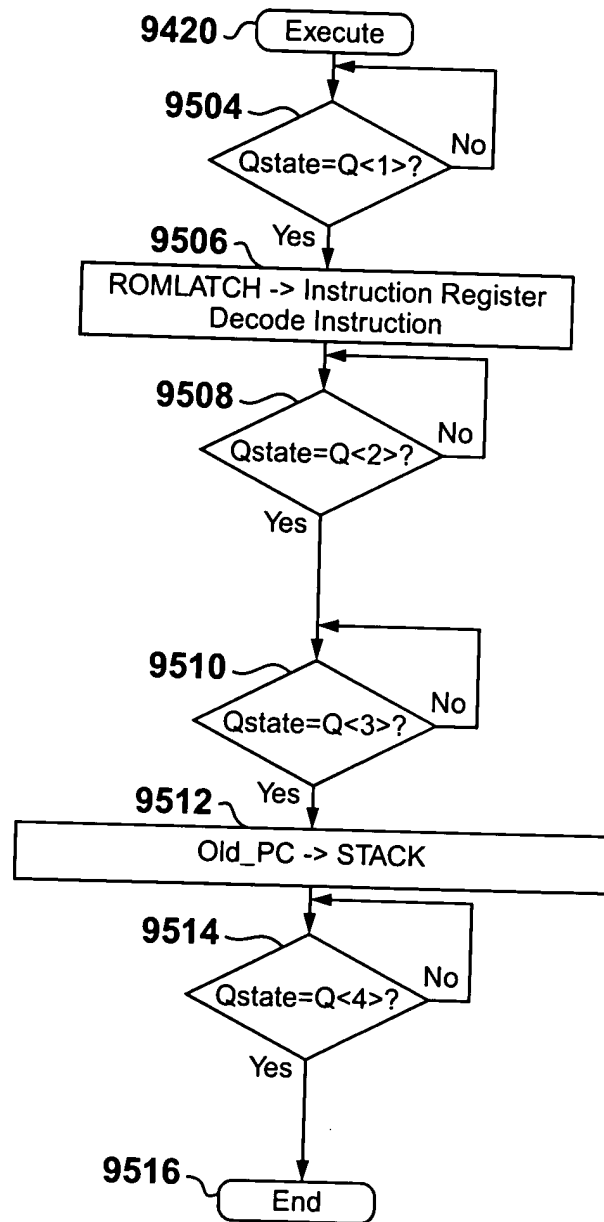


Figure 95

77/95

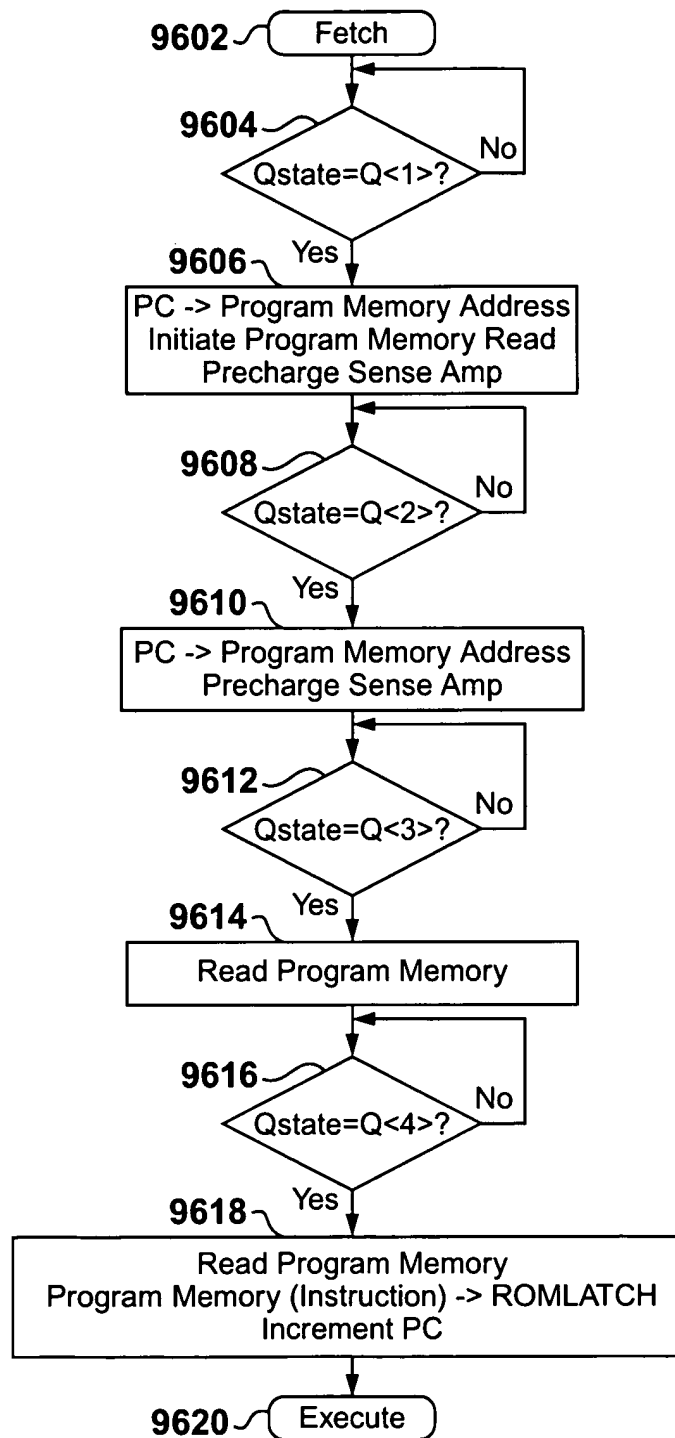


Figure 96

78/95

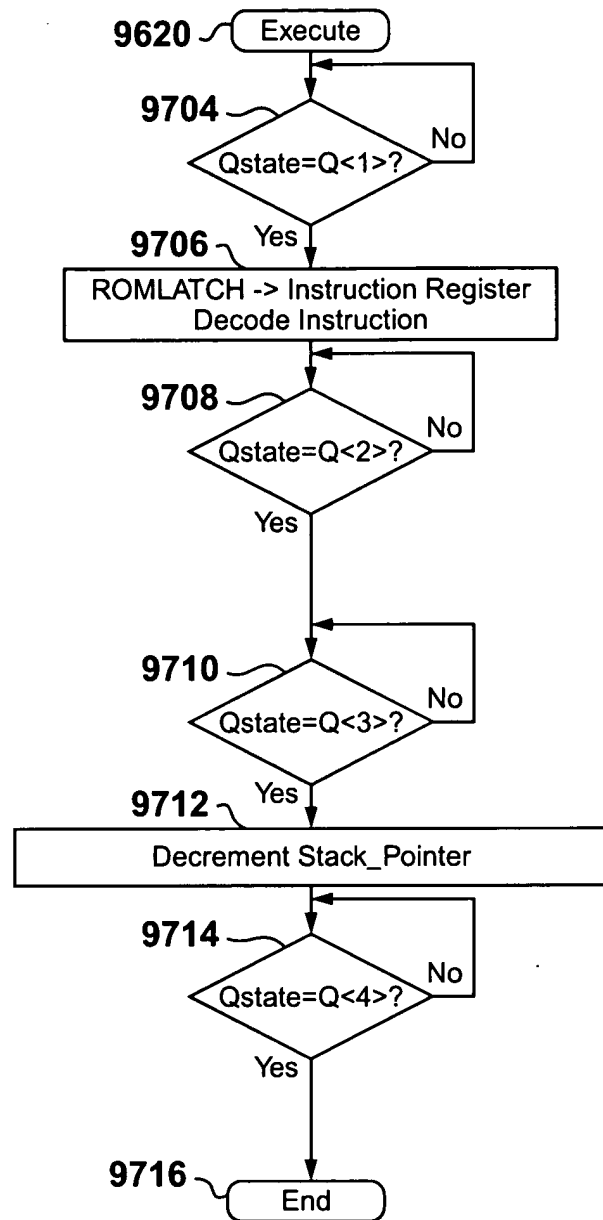


Figure 97

79/95

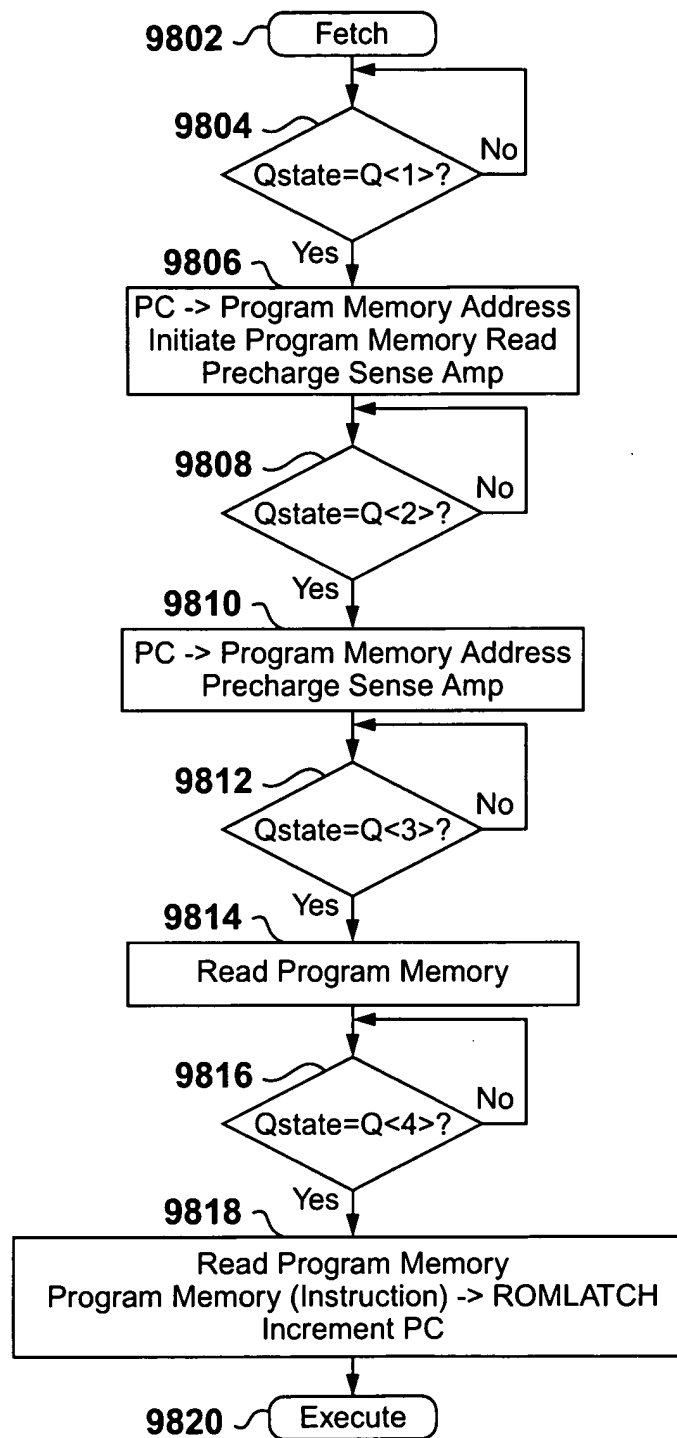


Figure 98

80/95

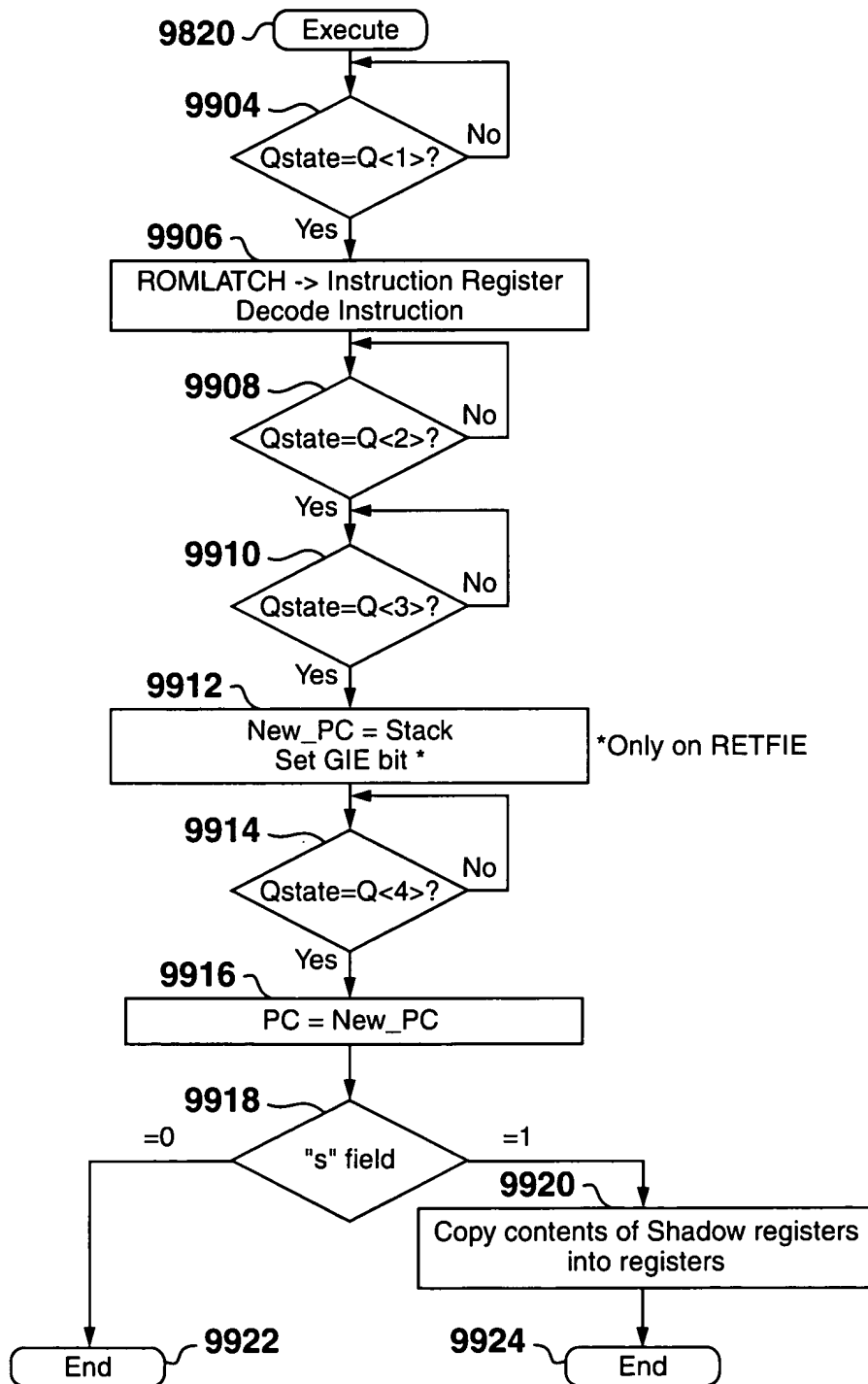


Figure 99

81/95

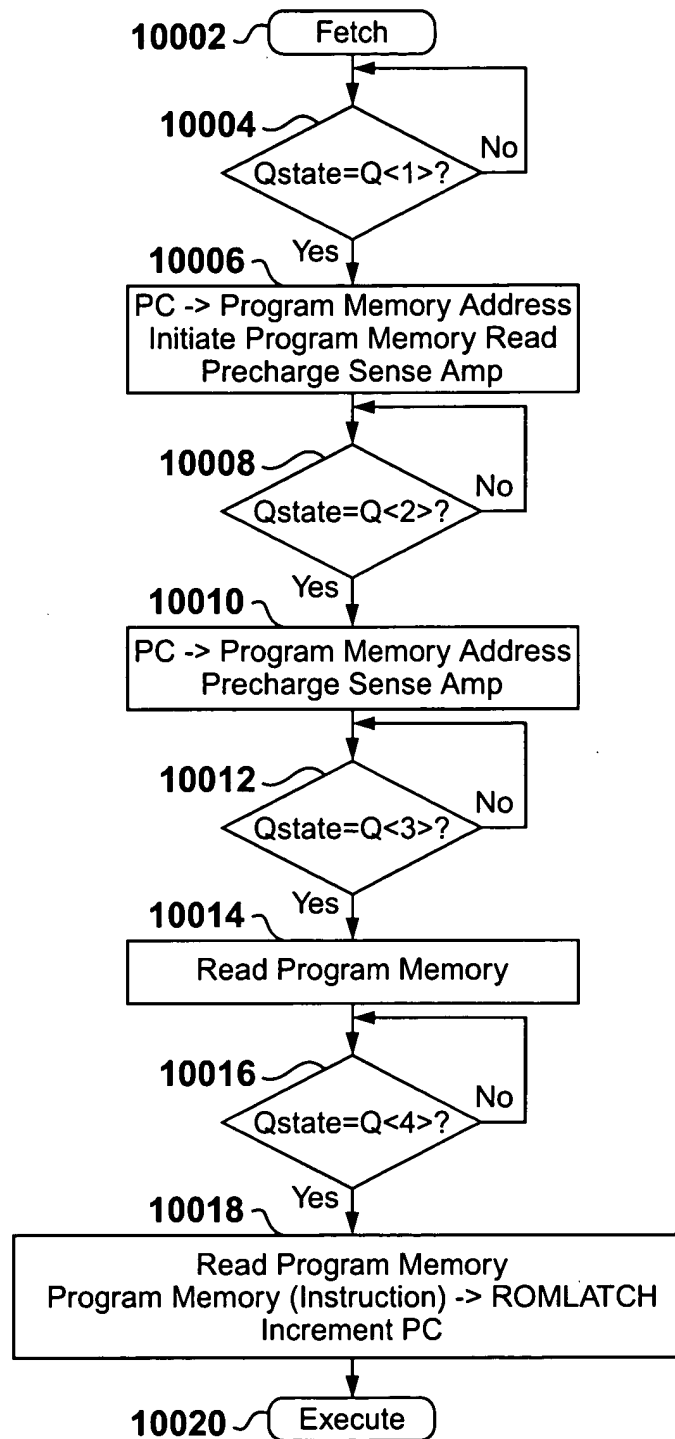


Figure 100

82/95

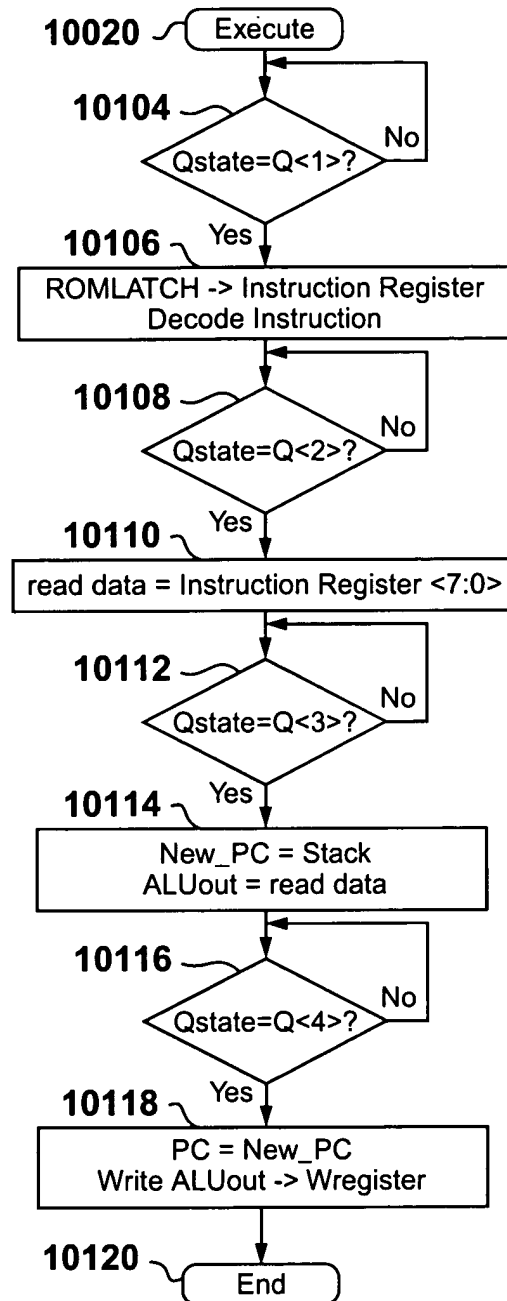


Figure 101

83/95

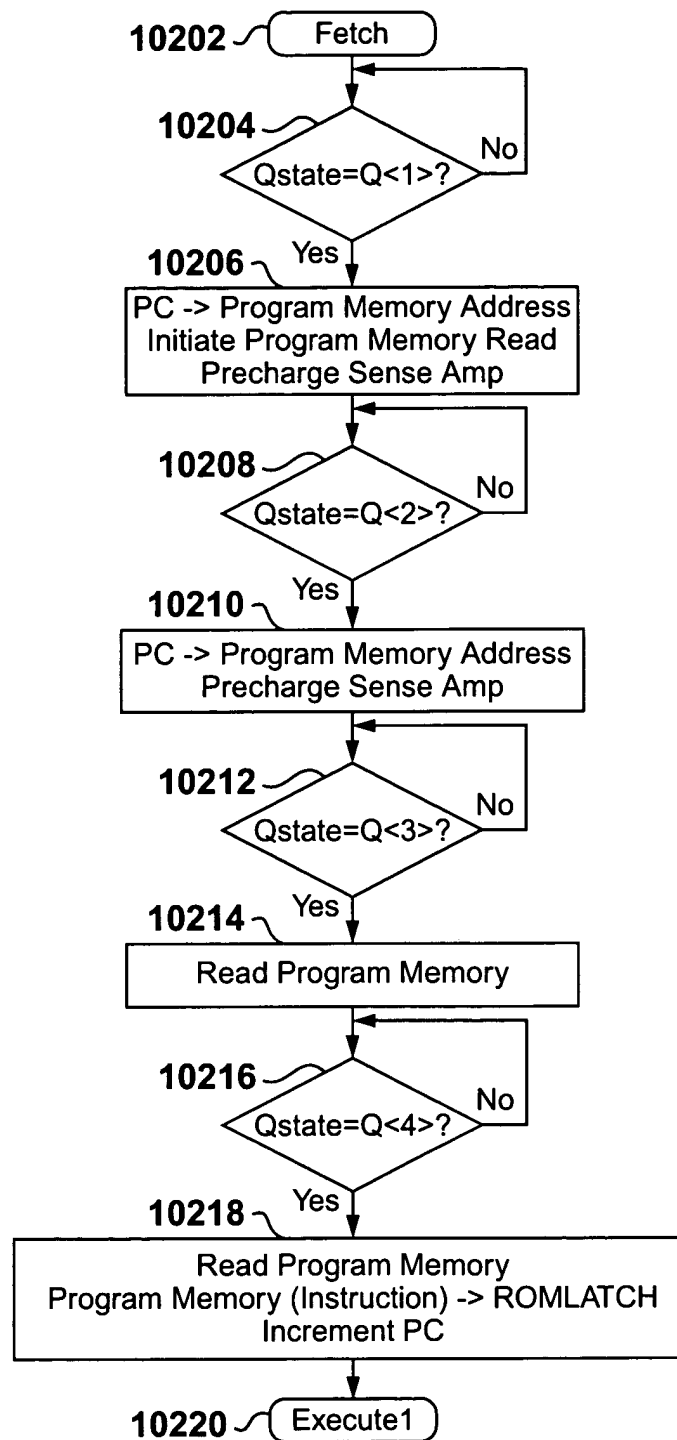
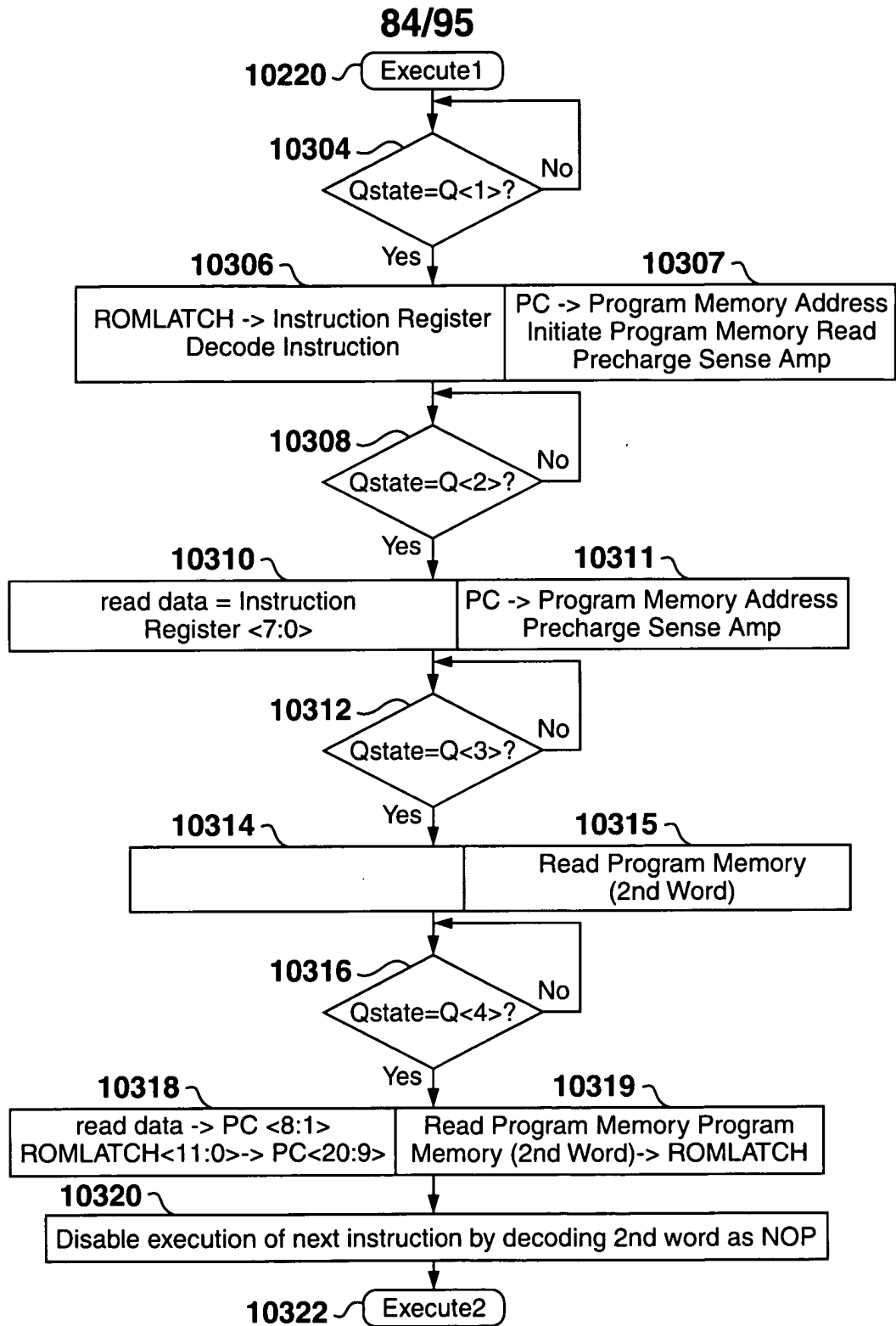


Figure 102



85/95

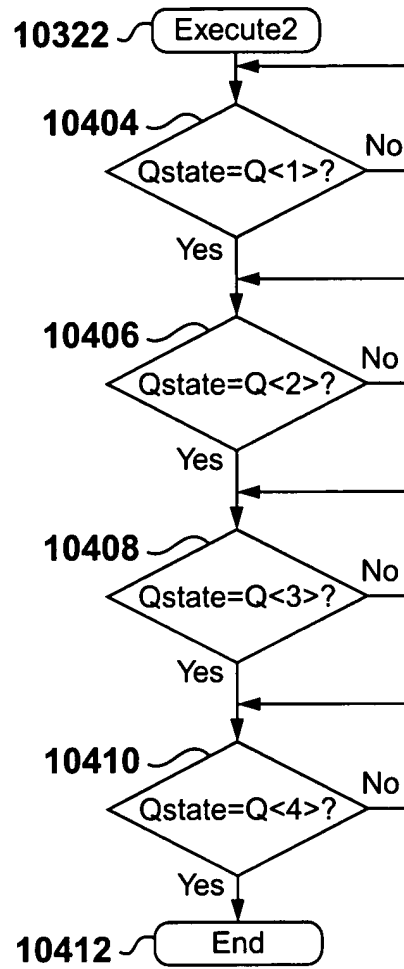


Figure 104

86/95

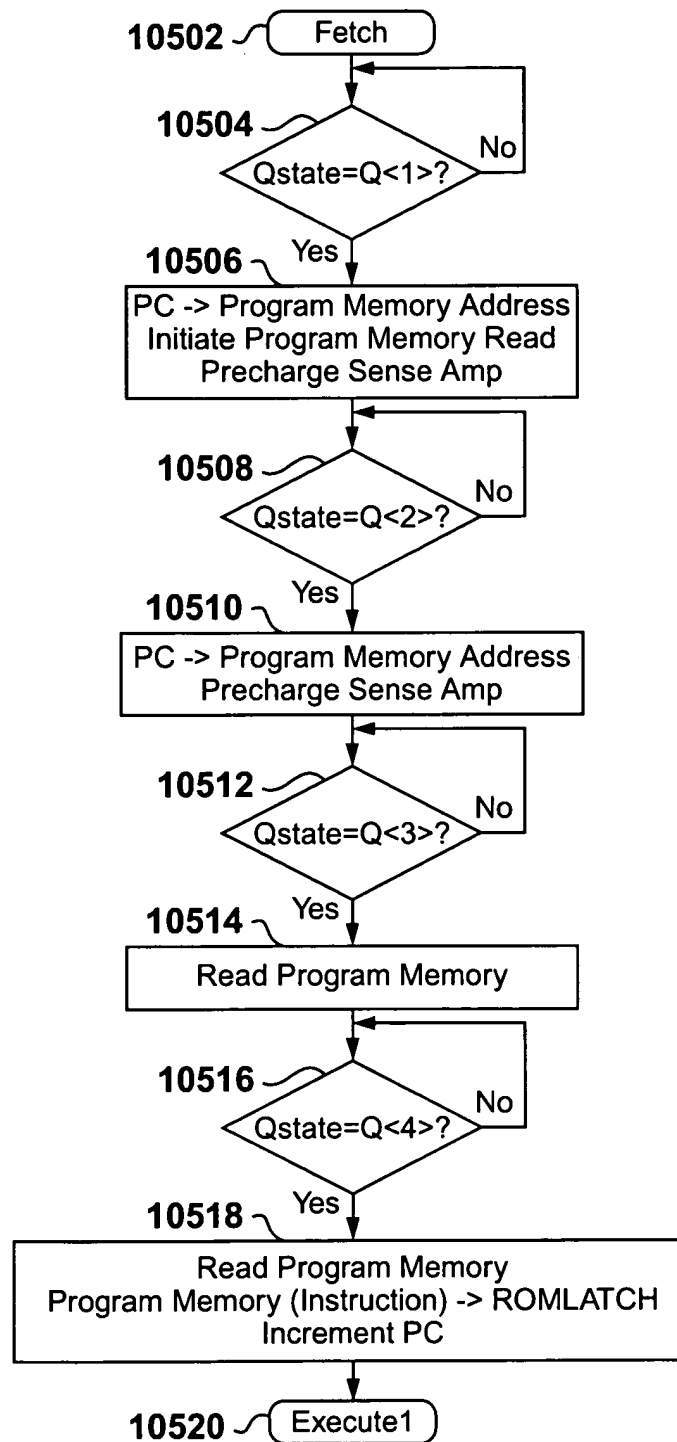


Figure 105

87/95

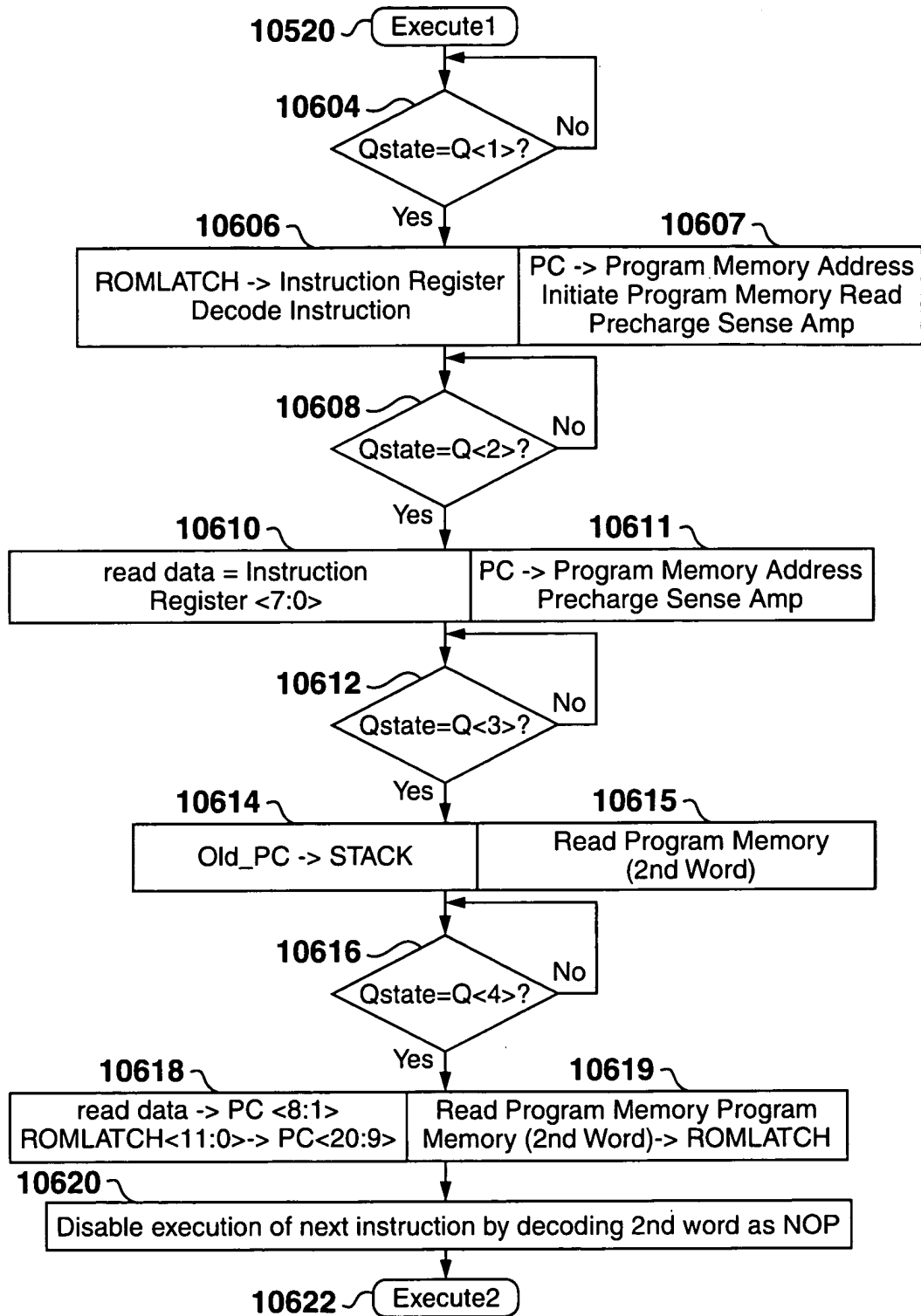


Figure 106

88/95

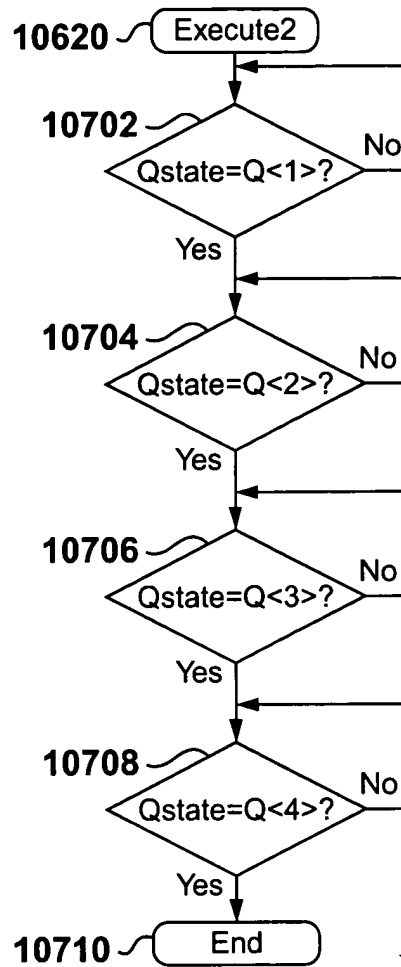


Figure 107

89/95

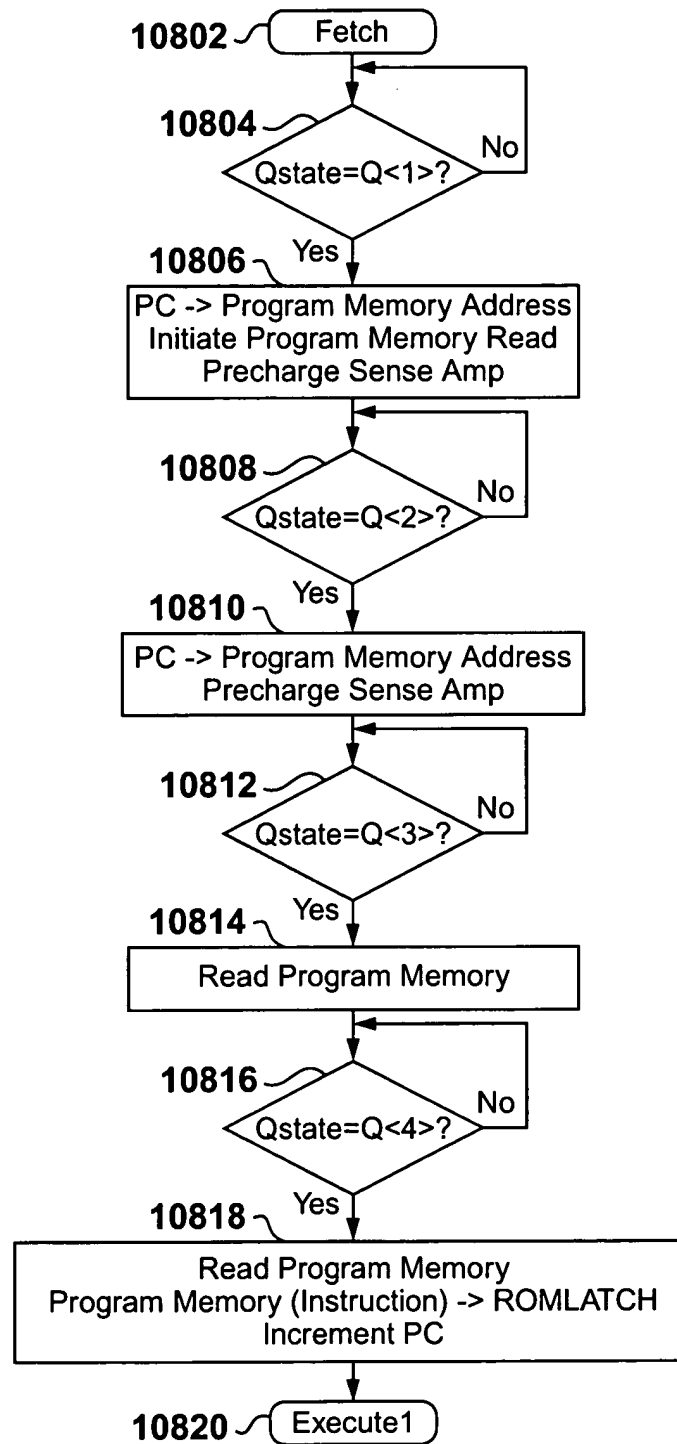


Figure 108

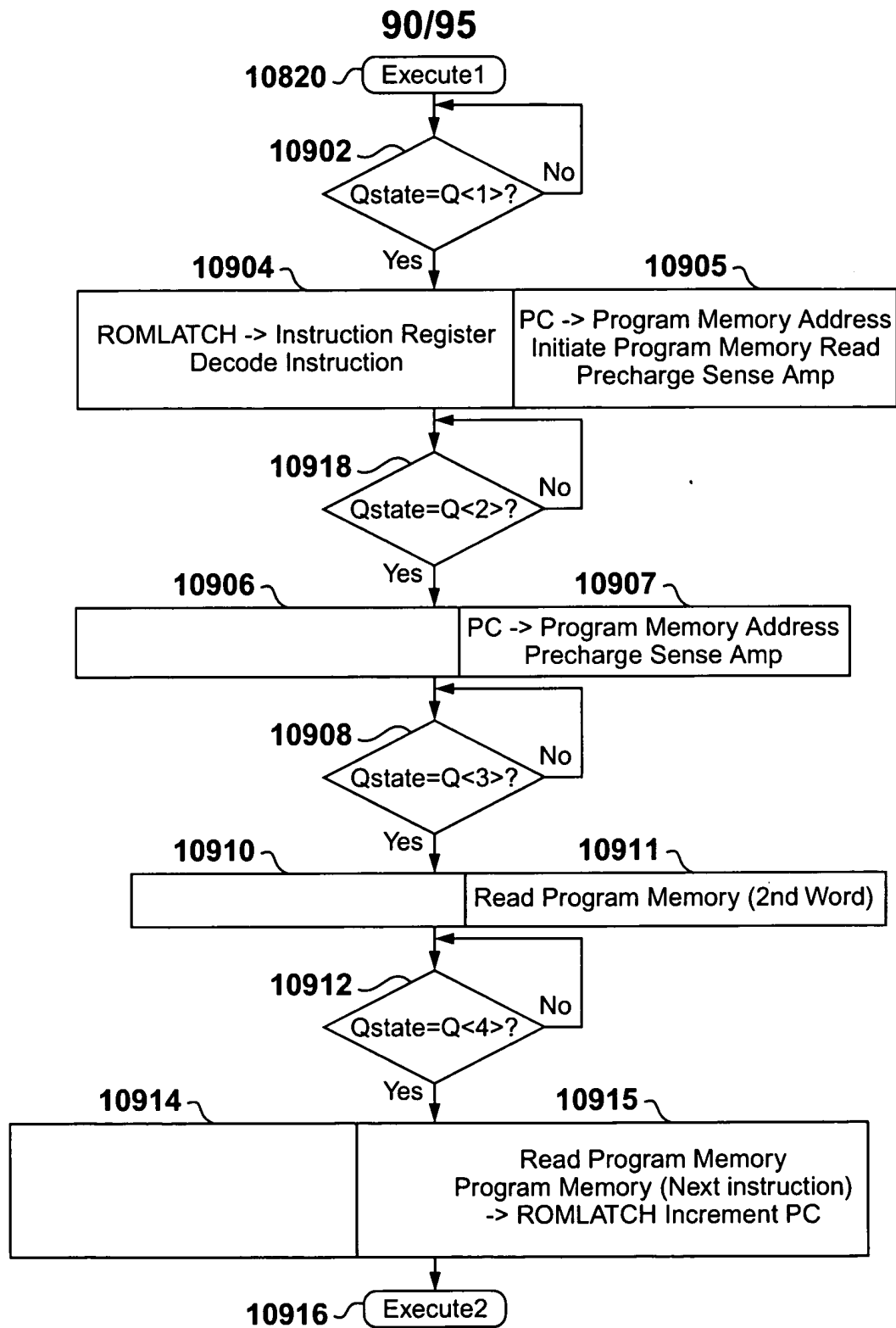


Figure 109

91/95

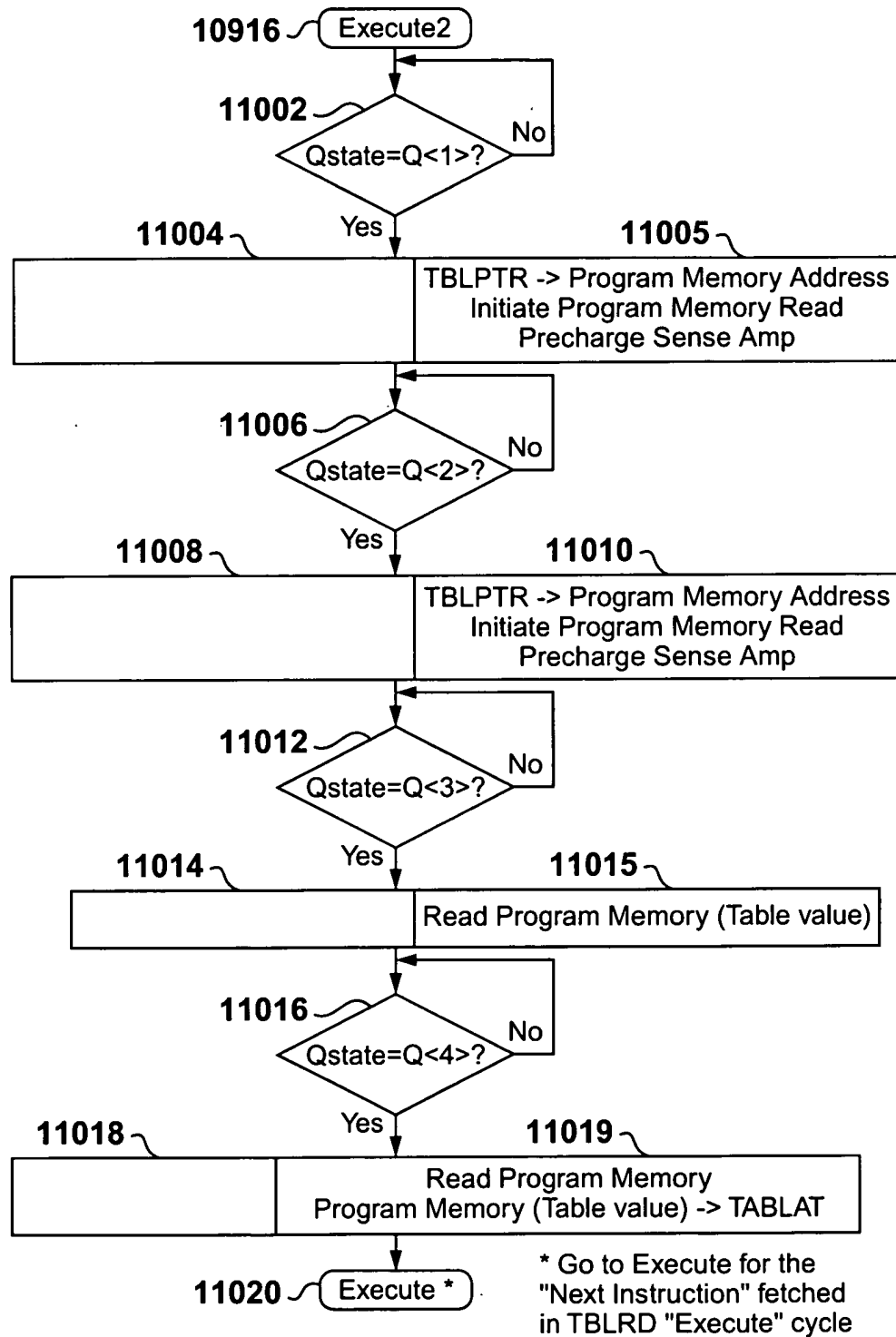


Figure 110

92/95

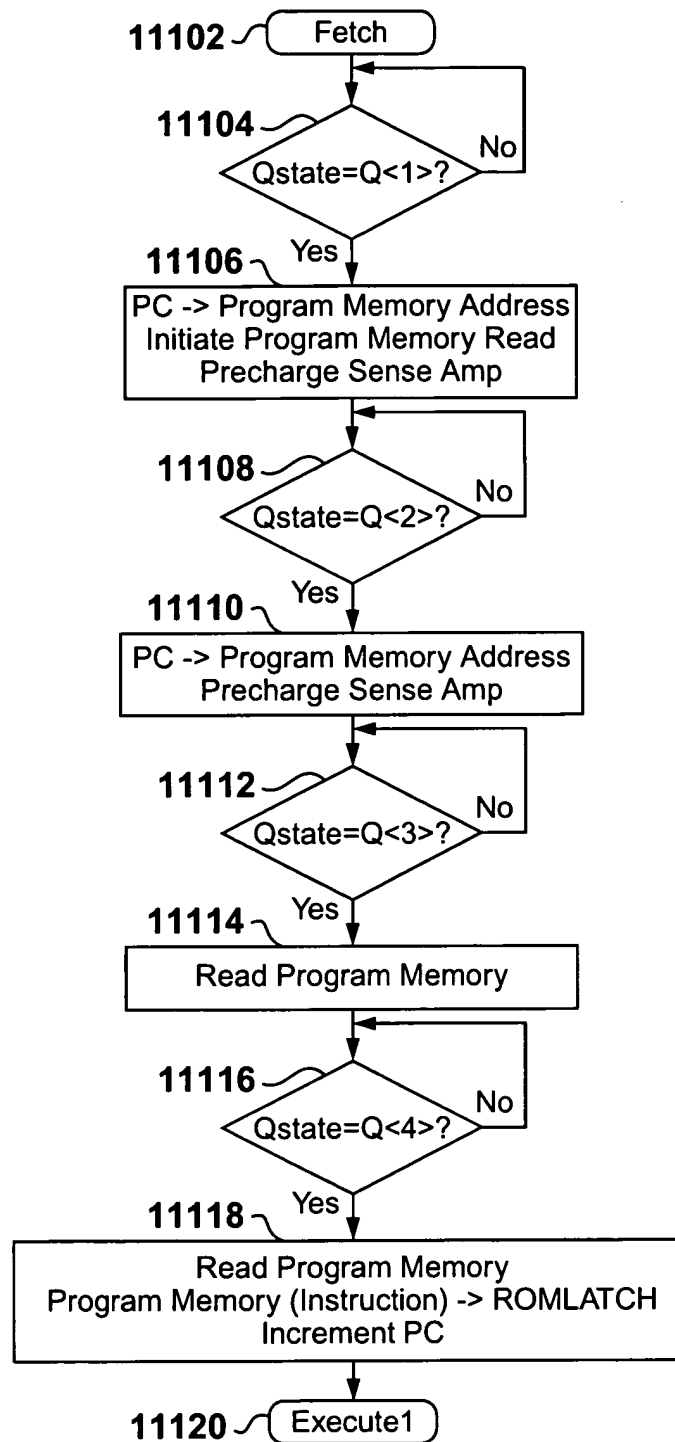


Figure 111

93/95

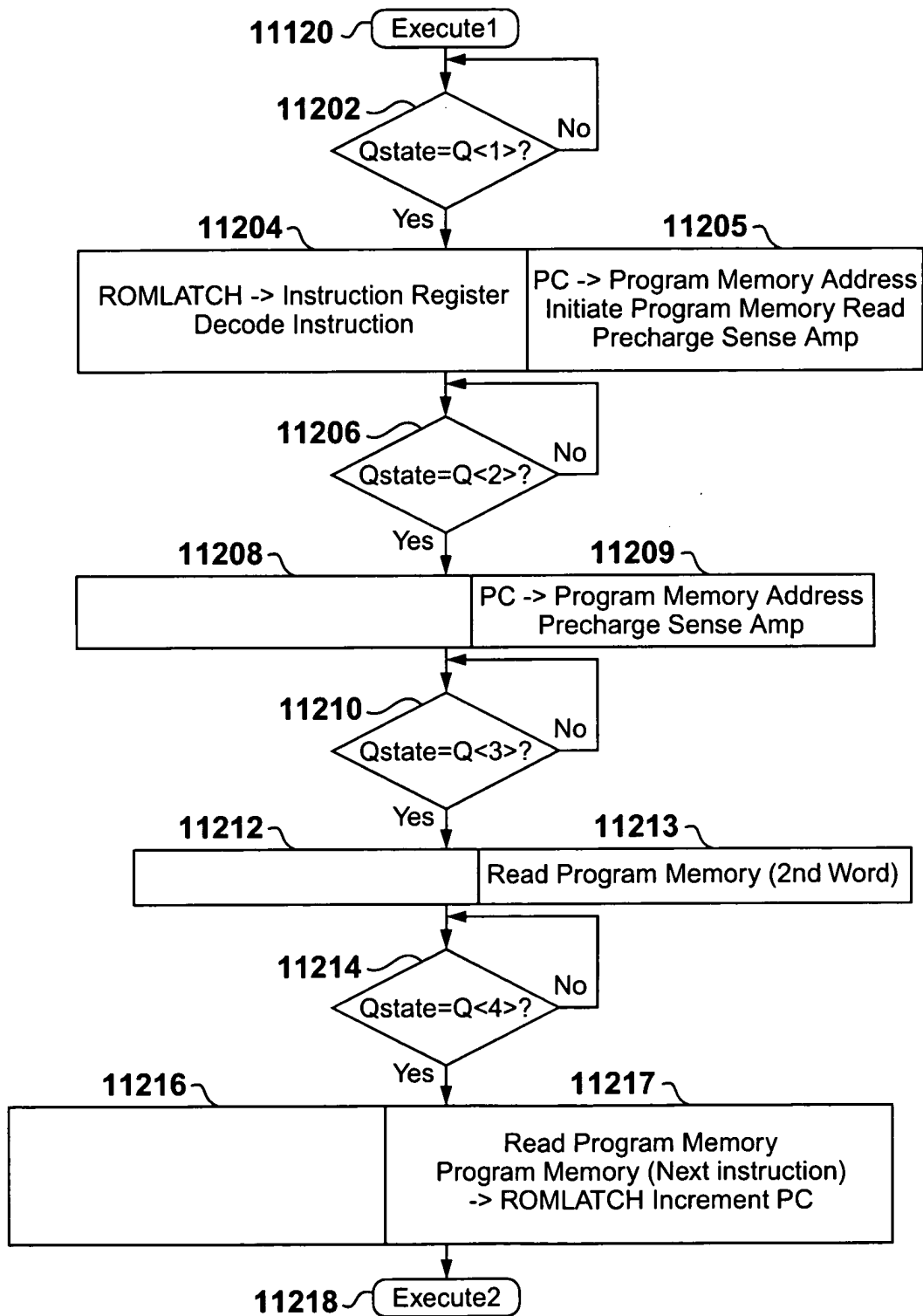


Figure 112

94/95

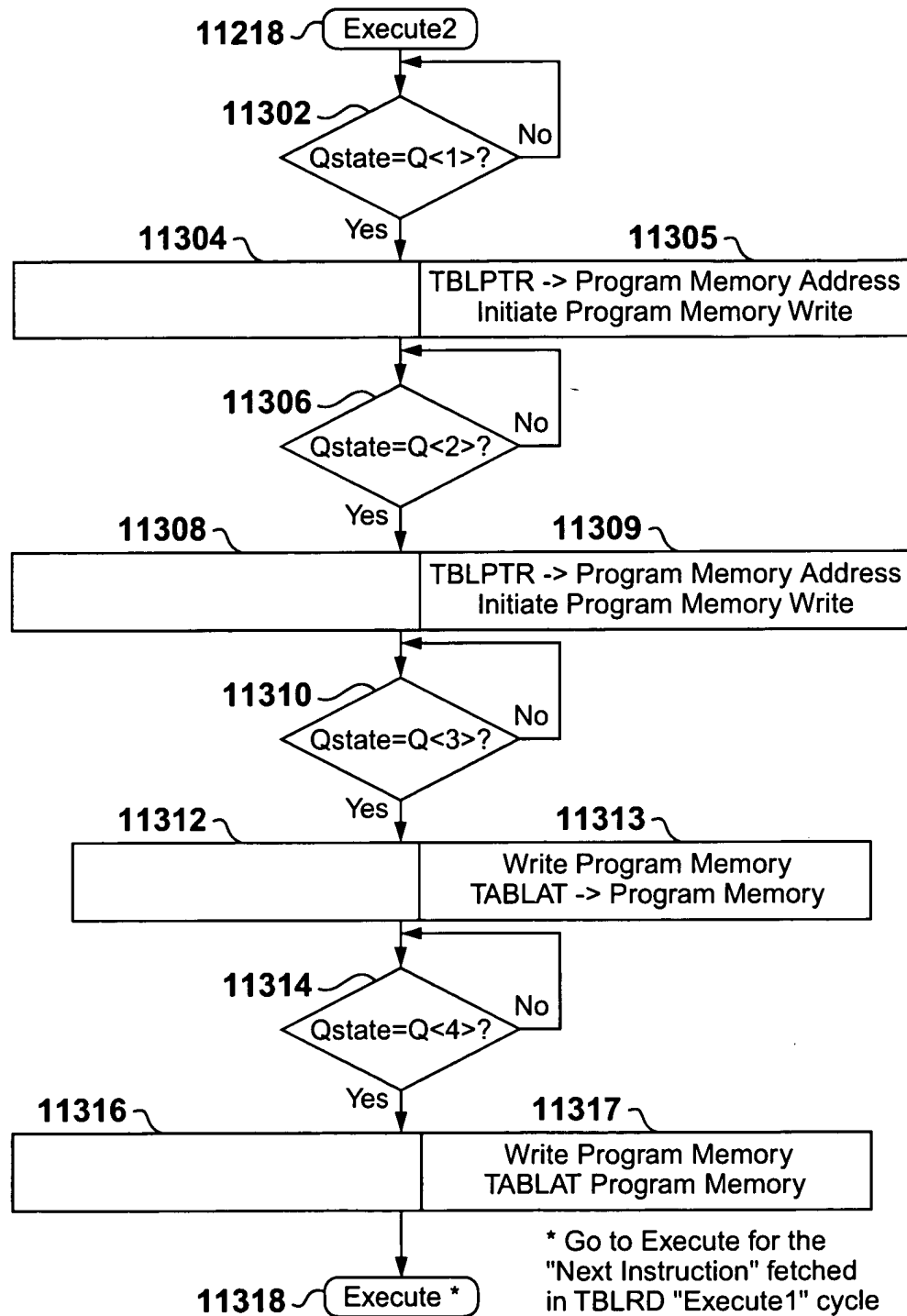


Figure 113

95/95

Opcode <11:8>

Opcode <15:12>

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	***	MOV LB	MULWF		DECF				SUB LW	IOR LW	XOR LW	AND LW	RET LW	MUL LW	MOV LW	ADD LW
1	IORWF				ANDWF				XORWF				COMF			
2	ADDWFC				ADDWF				INCF				DECFSZ			
3	RRCF				RLCF				SWAPF				INCFSZ			
4	RRNCF				RLNCF				INFSNZ				DCFSNZ			
5	MOVF				SUBFWB				SUBWFB				SUBWF			
6	CPFSLT		CPFSEQ		CPFSGT		TSTFSZ		SETF		CLRF		NEGF		MOVWF	
7	BTG															
8	BSF															
9	BCF															
A	BTFSS															
B	BTFSC															
C	MOVFF															
D	BRA								RCALL							
E	BZ	BNZ	BC	BNC	BV	BNV	BN	BNN	open				CALL		L FSR	GO TO
F	NOP (2ND WORD)															

***Special Instructions

0000	NOP
0001	HALT (NOTE: Emulation mode only.)
0003	SLEEP
0004	CLRWDT
0005	PUSH
0006	POP
0007	DAW
0008	TBLRD *
0009	TBLRD *+
000A	TBLRD *-
000B	TBLRD +*
000C	TBLWT *
000D	TBLWT *+
000E	TBLWT *-
000F	TBLWT +*
0010, 0011	RETFIE
0012, 0013	RETURN
00E0	TRAP
00E1	TRET
00FF	RESET

Figure 114